Performance Analysis with Vampir

Frank Winkler (frank.winkler@tu-dresden.de)
Performance tools will not automatically make your code run faster. They help you understand what your code does and where to put in work.
Agenda

Welcome to the Vampir Tool Suite
- Parallel Performance Analysis Approaches
- Mission
- Event Trace Visualization

The Vampir Workflow
- Score-P: Instrumentation & Run-Time Measurement
- Vampir & VampirServer

Vampir Performance Charts

Vampir Demo
- Tracing and Visualizing NPB-MZ-MPI / BT

Conclusions
Parallel Performance Analysis Approaches

Sampling
- Profile
  - Concise data sets
  - Good overview
  - Limited detail
  - No outliers

Instrumentation
- Event Trace
  - Extensive data sets
  - Most detailed
Mission

- Visualization of dynamics of complex parallel processes
- Requires two components
  - Monitor/Collector (Score-P)
  - Charts/Browser (Vampir)

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?
Event Trace Visualization with Vampir

- 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
Agenda

Welcome to the Vampir Tool Suite
- Parallel Performance Analysis Approaches
- Mission
- Event Trace Visualization

The Vampir Workflow
- Score-P: Instrumentation & Run-Time Measurement
- Vampir & VampirServer

Vampir Performance Charts

Vampir Hands-on
- Tracing and Visualizing NPB-MZ-MPI / BT

Conclusions
Score-P: Instrumentation & Run-Time Measurement

- Scalable Performance Measurement Infrastructure for Parallel Codes

- Supports a number of analysis tools
  - Periscope, Tau, Scalasca, Vampir

- Comes together with:
  - New Open Trace Format Version 2
  - CUBE4 profiling format
  - Opari2 instrumentor

- New BSD Open Source license
Score-P: Functionality

- Provide typical functionality for HPC performance tools

- Instrumentation (various methods)
  - Score-P compiler wrapper

- Flexible measurement without re-compilation:
  - Basic and advanced profile generation
  - Event trace recording
  - Online access to profiling data

- MPI, OpenMP, CUDA, and hybrid parallelism (and serial)
Score-P: Architecture

Vampir

Scalasca

TAU

Periscope

Event traces (OTF2)

Call-path profiles (CUBE4, TAU)

Score-P measurement infrastructure

Hardware counter (PAPI, rusage)

Application (MPI×OpenMP×CUDA)

Instrumentation wrapper

MPI

POMP2

CUDA

Compiler

TAU

User

PMPI

OPARI 2

CUDA

Compiler

PDT

User

Online interface
Score-P: Measurement Options

Measurements are configured via environment variables:

```
% scorep-info config-vars --full
```

- `SCOREP_ENABLE_PROFILING` 
  - Description: Total memory in bytes for the measurement system
- `SCOREP_ENABLE_TRACING` 
- `SCOREP_TOTAL_MEMORY` 
  - Description: Name of the experiment directory
- `SCOREP_EXPERIMENT_DIRECTORY` 
- `SCOREP_FILTERING_FILE` 
  - Description: A file name which contain the filter rules
- `SCOREP_METRIC_PAPI` 
  - Description: PAPI metric names to measure
- `SCOREP_METRIC_RUSAGE` 
  - Description: Resource usage metric names to measure

Profiles can be analyzed with `scorep-score`

- Helps to define appropriate filters for a tracing run
Vampir Tool Suite Workflow

1. Instrument your application with Score-P

   CC=icc
   CXX=icpc
   F90=ifc
   MPICC=mpicc

   CC=scorep icc
   CXX=scorep icpc
   F90=scorep ifc
   MPICC=scorep mpicc

2. Perform a measurement run with **profiling enabled**

3. Use **scorep-score** to define an appropriate filter

4. Perform a measurement run with **tracing enabled** and the filter applied

5. Perform in-depth analysis on the trace data with **Vampir**
Vampir – Visualization Modes (1)

Directly on front end or local machine

% vampir
On local machine with remote VampirServer

```
% vampirserver start -n 12
```

```
% vampir
```

---

VampirServer

Score-P

Many-Core Program

LAN/WAN

Large Trace File (stays on remote machine)

MPI parallel application
Agenda

Welcome to the Vampir Tool Suite
• Parallel Performance Analysis Approaches
• Mission
• Event Trace Visualization

The Vampir Workflow
• Score-P: Instrumentation & Run-Time Measurement
• Vampir & VampirServer

Vampir Performance Charts

Vampir Demo
• Tracing and Visualizing the NPB-MZ-MPI / BT

Conclusions
Main Charts of Vampir

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

Summary Charts:
- Function Summary
- Process Summary
- Communication Matrix View
Vampir: Charts for a WRF Trace with 64 Processes
Master Timeline
Process and Counter Timeline

![Timeline Diagram](image)
Agenda

Welcome to the Vampir Tool Suite
- Parallel Performance Analysis Approaches
- Mission
- Event Trace Visualization

The Vampir Workflow
- Score-P: Instrumentation & Run-Time Measurement
- Vampir & VampirServer

Vampir Performance Charts

Vampir Demo
- Tracing and Visualizing NPB-MZ-MPI / BT

Conclusions
The NAS Parallel Benchmark suite (MPI+OpenMP version)

- Available from: [http://www.nas.nasa.gov/Software/NPB](http://www.nas.nasa.gov/Software/NPB)
- 3 benchmarks in Fortran77 (bt-mz, lu-mz, sp-mz)
- Configurable for various sizes & classes (S, W, A, B, C, D, E)

Benchmark configuration for demo:

- Benchmark name: **bt-mz**
- Number of MPI processes: **NPROCS=4**
- Benchmark class: **CLASS=W**
- What does it do?
  - Solves a discretized version of unsteady, compressible Navier-Stokes equations in three spatial dimensions
  - Performs 200 time-steps on a regular 3-dimensional grid
Connect to Mira and add Vampir to the SoftEnv system

```bash
% vi .soft
  +vampir
% resoft
```

Copy sources to working directory

```bash
% cp /projects/Tools/vampir/tutorial/NPB3.3-MZ-MPI.tar.gz .
% tar xzvf NPB3.3-MZ-MPI.tar.gz
% cd NPB3.3-MZ-MPI
```

Compile the benchmark:

```bash
% make bt-mz CLASS=W NPROCS=4
cd BT-MZ; make CLASS=W NPROCS=4 VERSION=
makes: Entering directory 'BT-MZ'
cd ../sys; cc -o setparams setparams.c
../sys/setparams bt-mz 4 W
mpixlf77_r -c -O3 -qsmp=omp -qextname=flush bt.f
[...]
Built executable ../bin/bt-mz_W.4
make: Leaving directory 'BT-MZ'
```
Copy jobscript and launch as a hybrid MPI+OpenMP application

```bash
% cd bin
% cp ../jobscript/mira/run.sh .
% less run.sh
export OMP_NUM_THREADS=4
runjob -n 4 -p 4 --block $COBALT_PARTNAME --env-all : bt-mz_W.4
% qsub -A <projid> -t 10 -n 1 --mode script run.sh
% cat <jobid>.output

NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark
Number of zones:  4 x 4
Iterations:  200  dt:  0.000800
Number of active processes:  4
Total number of threads:  16  (  4.0 threads/process)

Time step 1
Time step 20
[...]  
Time step 200
Verification Successful

BT-MZ Benchmark Completed.
Time in seconds = 2.27
```

Hint: save the benchmark output (or note the run time) to be able to refer to it later
Edit `config/make.def` to adjust build configuration

Modify specification of compiler/linker: `MPIF77`

```bash
# SITE- AND/OR PLATFORM-SPECIFIC DEFINITIONS
#-------------------------------------------------------------------------------------
# Items in this file may need to be changed for each platform.
#-------------------------------------------------------------------------------------
...
#-------------------------------------------------------------------------------------
# The Fortran compiler used for MPI programs
#-------------------------------------------------------------------------------------
#MPIF77 = mpixlf77_r

# Alternative variants to perform instrumentation
...
MPIF77 = scorep mpixlf77_r

# This links MPI Fortran programs; usually the same as ${MPIF77}
FLINK   = $(MPIF77)
...
```

Uncomment the Score-P compiler wrapper specification
NPB-MZ-MPI / BT Instrumented Build

- Return to root directory and clean-up

```bash
% make clean
```

- Re-build executable using Score-P compiler wrapper

```bash
% make bt-mz CLASS=W NPROCS=4
cd BT-MZ; make CLASS=W NPROCS=4 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc -o setparams setparams.c
../sys/setparams bt-mz 4 W
scorep mpixlf77_r -c -O3 -qsmp=omp -qextname=flush bt.f

[...]
cd ../common; scorep mpixlf77_r -c -O3 -qsmp=omp -qextname=flush timers.f
scorep mpixlf77_r -O3 -qsmp=omp -qextname=flush -o ../bin.scorep/bt-mz_W.4
bt.o initialize.o exact_solution.o exact_rhs.o set_constants.o \
ad.o rhs.o zone_setup.o x_solve.o y_solve.o exch_qbc.o \
solve_subs.o z_solve.o add.o error.o verify.o mpi_setup.o \
../common/print_results.o ../common/timers.o
Built executable ../bin.scorep/bt-mz_W.4
make: Leaving directory 'BT-MZ'
```
Change to the directory containing the new executable before running it and adjust configuration

```bash
% cd bin.scorep
% cp ../jobscript/mira/* .
% less run_profile.sh
  export SCOREP_ENABLE_TRACING=false
  export SCOREP_ENABLE_PROFLING=true
  export SCOREP_TOTAL_MEMORY=100M
  export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt-mz_W_4x4_sum
  export OMP_NUM_THREADS=4
  runjob -n 4 -p 4 --block $COBALT_PARTNAME --env-all : bt-mz_W.4
% qsub -A <projid> -t 10 -n 1 --mode script run_profile.sh
% cat <jobid>.output
NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark
Number of zones: 4 x 4

[...]
Time step 200
Verification Successful

BT-MZ Benchmark Completed.
Time in seconds = 12.74
Creates experiment directory ./scorep_bt-mz_W_4x4_sum containing

- A record of the measurement configuration (scorep.cfg)
- The analysis report that was collated after measurement (profile.cubex)

```bash
% ls
... scorep_bt-mz_W_4x4_sum
% ls scorep_bt-mz_W_4x4_sum
profile.cubex scorep.cfg
```
Report scoring as textual output

% scorep-score scorep_bt-mz_W_4x4_sum/profile.cubex
Estimated aggregate size of event trace:
Estimated requirements for largest trace buffer (max_tbc):
(hint: When tracing set SCOREP_TOTAL_MEMORY > max_tbc to avoid intermediate flushes
or reduce requirements using file listing names of USR regions to be filtered.)

<table>
<thead>
<tr>
<th>flt type</th>
<th>max_tbc</th>
<th>time</th>
<th>% region</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>235123428</td>
<td>419.92</td>
<td>100.0 ALL</td>
</tr>
<tr>
<td>USR</td>
<td>232516724</td>
<td>78.19</td>
<td>18.6 USR</td>
</tr>
<tr>
<td>OMP</td>
<td>5973040</td>
<td>121.45</td>
<td>28.9 OMP</td>
</tr>
<tr>
<td>COM</td>
<td>314710</td>
<td>1.38</td>
<td>0.3 COM</td>
</tr>
<tr>
<td>MPI</td>
<td>88898</td>
<td>218.90</td>
<td>52.1 MPI</td>
</tr>
</tbody>
</table>

Region/callpath classification

- MPI (pure MPI library functions)
- OMP (pure OpenMP functions/regions)
- USR (user-level source local computation)
- COM (“combined” USR + OpenMP/MPI)
- ANY/ALL (aggregate of all region types)

868 MB total memory
224 MB per rank!
### Score report breakdown by region

```
% scorep-score -r scorep_bt-mz_W_4x4_sum/profile.cubex
[...]
flt type  max_tbc  time  % region
 ALL      235123428 419.92 100.0 ALL
 USER     232516724  78.19  18.6 USER
 OMP      5973040    121.45  28.9 OMP
 COM      314710     1.38   0.3 COM
 MPI      88898     218.90  52.1 MPI
[...]
```

More than 223 MB just for these 6 regions
Report scoring with prospective filter listing 6 USR regions

```
% cat ../config/scorep.filt
SCOREP_REGION_NAMES_BEGIN EXCLUDE
binvcrhs*
matmul_sub*
matvec_sub*
exact_solution*
binvrhs*
lhs*init*
timer_*

% scorep-score -f ../config/scorep.filt scorep_bt-mz_W_4x4_sum/profile.cubex
```

Estimated aggregate size of event trace: 20482398 bytes
Estimated requirements for largest trace buffer (max_tbc): 6377242 bytes
(hint: When tracing set SCOREP_TOTAL_MEMORY > max_tbc to avoid intermediate flushes or reduce requirements using file listing names of USR regions to be filtered.)

20 MB of memory in total, 6 MB per rank!
Available PAPI metrics

- Preset events: common set of events deemed relevant and useful for application performance tuning
  - Abstraction from specific hardware performance counters, mapping onto available events done by PAPI internally

```bash
% qsub -A <projid> -n 1 --mode c1 --proccount 1 -t 10 \
/soft/perftools/papi/bin/papi_avail
% cat <jobid>.outpout
```

- Native events: set of all events that are available on the CPU (platform dependent)

```bash
% qsub -A <projid> -n 1 --mode c1 --proccount 1 -t 10 \
/soft/perftools/papi/bin/papi_native_avail
% cat <jobid>.outpout
```

Note:
Due to hardware restrictions
- number of concurrently recorded events is limited
- there may be invalid combinations of concurrently recorded events
Re-run the application using the tracing mode of Score-P

```bash
% cd bin.scorep
% less run_trace.sh
    export SCOREP_ENABLE_TRACING=true
    export SCOREP_ENABLE_PROFLING=false
    export SCOREP_FILTERING_FILE=../config/scorep filt
    export SCOREP_TOTAL_MEMORY=100M
    export SCOREP_EXPERIMENT_DIRECTORY=scorep bt-mz_W_4x4_trace
    export SCOREP_METRIC_PAPI=PAPI_FP_OPS,PAPI_L1_DCM
    export OMP_NUM_THREADS=4
runjob -n 4 -p 4 --block $COBALT_PARTNAME --env-all : bt-mz_W.4
% qsub -A <projid> -t 10 -n 1 --mode script run_trace.sh
% cat <jobid>. output
NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark
Number of zones:  4 x  4
    [...]
Time step 200
Verification Successful

BT-MZ Benchmark Completed.
Time in seconds = 3.49
```
Download and install VampirClient for target platform

```bash
# Linux 64bit
$ scp <user>@mira.alcf.anl.gov:/projects/Tools/vampir/vampir-gui/vampir-*-x86_64.bin .
$ scp <user>@mira.alcf.anl.gov:/projects/Tools/vampir/vampir-gui/vampir-remote.license .
$ bash ./vampir-*.bin
```

Start VampirServer and follow output instructions

```bash
$ vampirserver start -a <projid> -n 16
Launching VampirServer...
Submitting PBS batch job (this might take a while)...
** Project 'tools'; job rerouted to queue 'prod-short'
VampirServer 8.2.1  (r8876)
Licensed to Mira Performance Boot Camp 2014
Running 15 analysis processes... (abort with vampirserver stop 28448)
VampirServer <28448> listens on: Q2G-I5-J01.mira.i2b:30066

Please run:
  ssh -L 30001:Q2G-I5-J01.mira.i2b:30066 <user>@mira.alcf.anl.gov
on your desktop to create ssh tunnel to VampirServer.

Start vampir on your desktop and choose 'Open Other -> Remote File'
  Description: mira, Server: localhost, Port: 30001
  Authentication: None
  Connection type: Socket
  Ignore "More Options"
```
Agenda

Welcome to the Vampir Tool Suite
• Parallel Performance Analysis Approaches
• Mission
• Event Trace Visualization

The Vampir Workflow
• Score-P: Instrumentation & Run-Time Measurement
• Vampir & VampirServer

Vampir Performance Charts

Vampir Demo
• Tracing and Visualizing NPB-MZ-MPI / BT

Conclusions
Conclusions

- Interactive trace visualization and analysis
- Intuitive browsing and zooming
- Scalable to large trace data sizes (20 TByte)
- Scalable to high parallelism (200,000 processes)
- Vampir for Linux, Windows and Mac OS

Score-P
- Common instrumentation and measurement infrastructure for various analysis tools
- Hides away complicated details
- Provides many options and switches for experts
Vampir is available at http://www.vampir.eu
Get support via vampirsupport@zih.tu-dresden.de
Score-P: http://www.vi-hps.org/projects/score-p