Session 3: Hands-On

Dr. Matthias S. Müller  (RWTH Aachen University)
Tobias Hilbrich       (Technische Universität Dresden)
Joachim Protze       (RWTH Aachen University, LLNL)

Email:
mueller@itc.rwth-aachen.de
tobias.hilbrich@tu-dresden.de
protze@itc.rwth-aachen.de
Content

- Options
- Setup
- MUST
- Vampir
Options

Local machine: **Chama**

Tools available:
- Score-P 1.2.3 (latest)
- Vampir 8.2 (very recent)
- MUST 1.2 (1.3 is latest)
- All configured for:
  openmpi-intel 1.6

```
% module load tools/vampir
```

VM:

- Easy-to-use environment
- Well tested
- Access via:
  - ssh
  - VNC
- IMPORTANT:
  - Not permanent, just today
  - Not managed by SNL/LANL
Setup – Virtual Machines, Overview

- We provide access to an Amazon EC2 based installation of Score-P, Vampir, and MUST
- You will need a laptop/desktop (Win/Linux/Mac)
- Open the following webpage:
  - http://rcswww.zih.tu-dresden.de/~hilbrich/trilabs14
  - http://tinyurl.com/trilabs14
- Goals:
  - Connect to the cloud
  - Trace a NAS Parallel Benchmark (bt)
  - Visualize the trace
  - Apply MUST
Receive a credential number (on cheat sheet)

Open: http://tinyurl.com/trilabs

We provide one virtual cluster for each participant exclusively

Use the cluster with your credential number!

There are multiple connection methods (details coming)
Setup – Virtual Machines, Credential Website

Debugging MPI and Hybrid-Heterogenous Applications at Scale

Material
- Slides (11MB)
- Heat conduction example: heat-errors.c, heat.h

Login Credentials for Hands-On

We use Amazon EC2 to run virtual machines as tiny clusters. Machines should provide 8 cores, where oversubscription is of possible, but impacts performance. We will boot one cluster per participant, the credentials for your cluster are listed in the table below. Please use the credentials of the row that matches the credential number that was assigned to you. You can either connect with VNC or a direct SSH login. VNC access is preferred as we use multiple graphical user interfaces.

VNC:
Option A (Linux): vncviewer [ip-address]:1
Option B (MAC): CotVNC (or any other VNC client, if installed)
Option C (Otherwise): Webbrowser Java plugin, enter the "VNC-Web" link into your Java enabled browser

For the ssh login issue:
ssh -X tools@[IP-ADDRESS]
X11 forwarding may be slow you should prefer a VNC connection for any GUI usage.

<table>
<thead>
<tr>
<th>#</th>
<th>IP-Address</th>
<th>VNC-Web</th>
<th>Password</th>
<th>SSH Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50.18.72.181</td>
<td><a href="http://50.18.72.181:5801">http://50.18.72.181:5801</a></td>
<td>sc12sc12</td>
<td>ssh -X tools@50.18.72.181</td>
</tr>
</tbody>
</table>

Connection details (coming)

Credential table

One error in opening the page. For more information, choose Window > Activity.
Setup – Virtual Machines, VNC

- Helps since we use multiple GUls
  - X11 forwarding can become slow
- Option A (Linux): vncviewer <ip>:1
- Option B (Mac): CotVNC (if you have it)
- Option C (Windows + Mac + [Mobile]):
  - Credential table: Use “VNC-Web” link (Requires Java)
- You enter a simple X11 environment (xfce)
- To open a shell (one available when you log in)
  - Main Menu -> Terminal
Setup – Virtual Machines, SSH Connection

- As fallback use direct login to your VM instance
- Enable X11 forwarding (-X) and compression (-C):
  ```
  % ssh -XC tools@<IP>
  ```
- Password is in the credential table (trilabs14)
- Vampir GUI may be slow
Setup – Virtual Machines, Directory Structure

```plaintext
% cd $HOME
% ls

- manuals
  - Manuals for all tools
- must-training
  - Material for MUST training
- NPB3.3-MPI
  - Score-P tracing example
- heat-example
  - Heat conduction example (contains a series of errors)
```
MUST

• Go into the NPB directory
• Edit config/make.def
• Disable any other tool (i.e. use mpif77/mpiifort)
  – For Bluegene use mustf77 linking wrapper
• Build:

% make bt NPROCS=4 CLASS=B
===========================================
=      NAS PARALLEL BENCHMARKS 3.3       =
=      MPI Versions        =
=      F77               =
===========================================

cd BT-MZ; make CLASS=B NPROCS=4
make[1]: Entering directory
...

mpif77 -g -O2 -o ../bin/bt-mz.B.4 bt.o initialize.o ...
make[1]: Leaving directory
• Go to bin directory

% cd bin

• Run with tool:

% mustrun -np 4 ./bt.W.4
Output should read like:

```bash
% cd bin
% mustrun --must:nocrash -np 4 bt.W.4
  Weaver ... success
Code generation ... success
Build file generation ... success
Configuring intermediate build ... success
Building intermediate sources ... success
Installing intermediate modules ... success
Generating P^nMPI configuration ... success
Search for preloaded P^nMPI ... not found ... success
Executing application: NAS Parallel Benchmarks 3.3 -- BT Benchmark
... Iterations: 200  dt: 0.0008000
  Number of active processes: 4

Time step 1
...
Verification Successful
```
Local Access – Visualization with Remote Analysis

**Step 1:** Start VampirServer (% vampirserver start -n X)

**Step 2:** Portforwarding (% ssh -L ...)

**Step 3:** Install and start GUI (% vampir)

Analysis: VampirServer

TCP Socket connection

Visualization: Vampir
Local Access – Remote Analysis, VampirServer

- If necessary, load modules for vampir/vampirserver

% module load vampirserver
Vampirserver 8.2 loaded

- Start VampirServer
  - If configured by the admin, this submits a batch job!

% vampirserver start -n 31
Launching VampirServer...
Submitting LSF batch job (this might take a while)...
VampirServer 8.2.0  (r8690)
Licensed to VI-HPS Tools Workshop 02/2014
Running 31 analysis processes... (abort with vampirserver stop 8921)
VampirServer <8921> listens on: s04r1b78:30075

Remember these coordinates, you need them later
Local Access – Remote Analysis, Port Forwarding

Note the compute node on which the server runs:

```
% vampirserver start -n 31
Launching VampirServer...
Submitting LSF batch job (this might take a while)...
VampirServer 8.2.0 (r8690)
Licensed to VI-HPS Tools Workshop 02/2014
Running 31 analysis processes... (abort with vampirserver stop 8921)
VampirServer <8921> listens on: s04r1b78:30075
```

Establish port forwarding from your local machine:

```
% ssh \
  -L 30000:s04r1b78:30075 \ 
  <user>@<machine>
```
Local Access – Remote Analysis, GUI

On your local machine, start vampir:

```
vampir
```

Use the “Open Other” option.
Local Access – Remote Analysis, GUI (2)

Select “Remote File”
Local Access – Remote Analysis, GUI (3)

Server is “localhost”

Connection type “Socket”

Port is “30000”
Local Access – Remote Analysis, GUI (4)
Local Access – Remote Analysis, GUI (5)

Master Timeline

Navigation Toolbar

Function Summary

Function Legend
Tracing – Usage

Compiler pre-command provides instrumentation:

- CC=gcc
- CXX=g++
- F90=gfortran
- MPICC=mpicc

- CC=scorep gcc
- CXX=scorep g++
- F90=scorep gfortran
- MPICC=scorep mpicc

Basic Score-P usage:
- Put “scorep” in front of compiler commands
- Re-compile & re-link
- Run as usual (details follow)

User function instrumentation: compiler-based
Score-P measurements are configured via environment variables:

```
% scorep-info config-vars --full
SCOREP_ENABLE_PROFILING
   Description: Enable profiling
       [...]
SCOREP_ENABLE_TRACING
   Description: Enable tracing
       [...]
SCOREP_TOTAL_MEMORY
   Description: Total memory in bytes for the measurement system
       [...]
SCOREP_experiment_directory
   Description: Name of the 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
       [... More configuration variables ...]
```
NAS Parallel Benchmarks

- Activate Score-P for compilation:

```bash
% vi config/make.def

#MPIF77 = mpif77
MPIF77 = scorep mpif77
```

- Build a kernel:

```bash
% make bt CLASS=W NPROCS=4
```
Experiment 1: Profile run

% cd bin
% export SCOREP_ENABLE_PROFILING=1
% export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt_profile
% mpirun -np 4 bt.W.4

NAS Parallel Benchmarks (NPB3.3-MZ-MPI)
- BT-MZ MPI+OpenMP Benchmark
Size: 24x 24x 24
Iterations: 200  dt: 0.0008000
Number of active processes: 4

Time step 1
Time step 20
[...]
Time step 180
Time step 200
Verification Successful

BT-MZ Benchmark Completed.
Time in seconds = 11.17
Tracing – Example, Inspect Profiles

- Creates directory `.scorep_bt_profile` containing:
  - A record of the measurement configuration (`scorep.cfg`)
  - The analysis report that was collated after measurement (`profile.cubex`)

```
% ls
... scorep_bt_profile
% ls scorep_bt_profile
profile.cubex scorep.cfg
```

- Interactive exploration with CUBE/ParaProf

```
% cube scorep_bt_profile/profile.cubex
[CUBE GUI showing summary analysis report]

% paraprof scorep_bt_profile/profile.cubex
[TAU ParaProf GUI showing summary analysis report]
```
**Tracing – Example, Prepare for Tracing**

- **Report tracing information as textual output**

```%
-scorep-scorep_bt_profile/profile.cubex
```

Estimated aggregate size of event trace (total_tbc):
Estimated requirements for largest trace buffer (max_tbc):
(hint: 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>103800930</td>
<td>49.71</td>
<td>100.0</td>
</tr>
<tr>
<td>USR</td>
<td>103389320</td>
<td>44.41</td>
<td>89.3</td>
</tr>
<tr>
<td>MPI</td>
<td>343150</td>
<td>5.14</td>
<td>10.3</td>
</tr>
<tr>
<td>COM</td>
<td>68460</td>
<td>0.15</td>
<td>0.3</td>
</tr>
</tbody>
</table>

**Region/callpath classification**

- MPI (pure MPI library functions)
- OMP (pure OpenMP functions/regions)
- COM (“combined” USR + OpenMP/MPI)
- ANY/ALL (aggregate of all region types)
Report breakdown by region

```bash
% scorep-score -r scorep_bt_profile/profile.cubex
[...]
flt type    max_tbc       time      % region
ALL         103800930    49.71     100.0 ALL
USR         103389320    44.41     89.3 USR
            343150       5.14      10.3 MPI
            68460        0.15      0.3 COM
USR         33562980     7.21      14.5 matmul_sub_
USR         33562980     7.79      15.7 binvcrhs_
USR         33562980     6.62      13.3 matvec_sub_
USR         1459260      0.31      0.6 binvrhs_
USR         1136400      0.24      0.5 exact_solution_
MPI         145080       0.05      0.1 MPI_Isend
MPI         145080       0.03      0.1 MPI_Irecv
MPI         48240        0.67      1.4 MPI_Wait
USR         24120        0.01      0.0 lhsabinit_
```

Most trace buffer space used for these 5 regions
Create and evaluate a filter:

```
% nano filter.txt
SCOREP_REGION_NAMES_BEGIN
  EXCLUDE
    matmul_sub*
    binvcrhs*
    matvec_sub*
    binvrhs*
    exact_solution*
SCOREP_REGION_NAMES_END

% scorep-score -f filter.txt scorep_bt_profile/profile.cubex
Estimated aggregate size of event trace (total_tbc):
  2065320 bytes
...`

With filter: 2 MB of memory in total
Tracing – Example, Tracing Run

Experiment 2: Tracing with filter

% export SCOREP_ENABLE_PROFILING=0
% export SCOREP_ENABLE_TRACING=1
% export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt_trace
% export SCOREP_FILTERING_FILE=filter.txt
% mpirun -np 4 bt.A.4

NAS Parallel Benchmarks 3.3 -- BT Benchmark
Size: 24x 24x 24
Iterations: 200 dt: 0.0008000
Number of active processes: 4

Time step 1
Time step 20
[...]
Time step 180
Time step 200
Verification Successful

BT-MZ Benchmark Completed.
Time in seconds = 1.34
Visualization: Most simple (Analysis on Frontend)

% ssh -XC <user>@<machine>
% module load vampir
% vampir <scorep-trace-directory>/traces.otf2
Tracing – Visualization Example

% vampir scorep_bt_trace/traces.otf2

High MPI ratio due to 4 ranks on 2 cores
Tracing – Adding Detail

- We traced:
  - Function calls
  - MPI calls

- We might add (examples):
  - More buffering for the trace (for longer runs):
    \[ \Rightarrow \% \text{ export } \text{SCOREP\_TOTAL\_MEMORY}=100M \]
  - PAPI performance counters (not on the VM):
    \[ \Rightarrow \% \text{ export } \text{SCOREP\_METRIC\_PAPI}=\text{PAPI\_FP\_OPS} \]
  - CUDA events via the CUPTI interface:
    \[ \Rightarrow \% \text{ export } \text{SCOREP\_CUDA\_ENABLE}=\text{gpu, kernel, idle} \]
Tracing – Compiler Instrumentation Limits

- Default instrumentation: compiler instrumentation
  - Instruments EVERY function call
  - Even with filter: Score-P invoked per function call
  - Can cause high overhead

- More advanced instrumentation techniques:
  - Manual (laborious, but always available)
  - Source-to-source instrumentation with TAU PDT
  - Legacy VampirTrace: Dyninst binary instrument.
  - Score-P upcoming: Sampling + Tracing

- Advice: If you observe considerable perturbation, apply a different function instrumentation technique
Tracing – Manual / PDT Instrumentation

Instrumentation and execution with PDT:

Activates PDT instrumentation (Optional Score-P feature)

% scorep --pdt mpicc source.c -o exe
% mpirun -np 4 ./exe

Manual instrumentation:

Activates user instrumentation (actual instrumentation in the source files)

% scorep --user mpicc source.c -o exe
% mpirun -np 4 ./exe

Upcoming in Score-P: Sampling
• Requires processing by the C preprocessor
#include "scorep/SCOREP_User.h"

void foo()
{
    /* Declarations */
    SCOREP_USER_REGION_DEFINE( solve )

    /* Some code... */
    SCOREP_USER_REGION_BEGIN( solve, "<solver>", \ 
                              SCOREP_USER_REGION_TYPE_LOOP )
    for (i = 0; i < 100; i++)
    {
        [...]
    }
    SCOREP_USER_REGION_END( solve )
    /* Some more code... */
}
#include "scorep/SCOREP_User.h"

void foo()
{
    // Declarations

    // Some code...
    {
        SCOREP_USER_REGION( "<solver>" , SCOREP_USER_REGION_TYPE_LOOP )
        for (i = 0; i < 100; i++)
        {
            [...] 
        }
    }
    // Some more code...
}
Can be used to temporarily disable measurement for certain intervals

- Annotation macros ignored by default
- Enabled with `[--user]` flag

```fortran
#include "scorep/SCOREP_User.inc"

subroutine foo(...)  
  ! Some code...  
  SCOREP_RECORDING_OFF()  
  ! Loop will not be measured  
  do i=1,100  
    [...]  
    end do  
  SCOREP_RECORDING_ON()  
  ! Some more code...  
end subroutine
```

```c
#include "scorep/SCOREP_User.h"

void foo(...) {  
  /* Some code... */  
  SCOREP_RECORDING_OFF()  
  /* Loop will not be measured */  
  for (i = 0; i < 100; i++) {  
    [...]  
  }  
  SCOREP_RECORDING_ON()  
  /* Some more code... */  
}
```

Fortran (requires C preprocessor)  

```c
C / C++
```