Vampir hands-on: Visualizing and analyzing NPB-MZ-MPI / BT
Demo example code:
NPB-MZ-MPI / BT
NPB-MZ-MPI suite

- The NAS Parallel Benchmark suite (MPI+OpenMP version)
  - Available from: http://www.nas.nasa.gov/Software/NPB
  - 3 benchmarks in Fortran77
  - Configurable for various sizes & classes
- Move into the NPB3.3-MZ-MPI root directory

```
% cd tutorial; ls
bin/   common/ jobscript/ Makefile README.install SP-MZ/
BT-MZ/ config/ LU-MZ/ README README.tutorial sys/
```

- Subdirectories contain source code for each benchmark
  - plus additional configuration and common code
- The provided distribution has already been configured for the tutorial, such that it's ready to “make” one or more of the benchmarks and install them into a (tool-specific) “bin” subdirectory
NPB-MZ-MPI / BT (Block Tridiagonal solver)

- 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
  - Implemented in 20 or so Fortran77 source modules

- Uses MPI & OpenMP in combination
  - 4 processes with 4 threads each should be reasonable
    - don’t expect to see speed-up when run on a laptop!
  - bt-mz_W.4 should run in around 13 seconds on a laptop
  - bt-mz_C.4 is more suitable for dedicated HPC compute nodes
    - Each class step takes around 10-15x longer
Building an NPB-MZ-MPI benchmark

- Type “make” for instructions

```make
% make
===========================================
= NAS PARALLEL BENCHMARKS 3.3 =
= MPI+OpenMP Multi-Zone Versions =
= F77 =
===========================================

To make a NAS multi-zone benchmark type

```make
make <benchmark-name> CLASS=<class> NPROCS=<nprocs>
```

where `<benchmark-name>` is “bt-mz”, “lu-mz”, or “sp-mz”
`<class>` is “S”, “W”, “A” through “F”
`<nprocs>` is number of processes

[...]

**************************************************************
* Custom build configuration is specified in config/make.def *
* Suggested tutorial exercise configuration for LiveISO/DVD: *
* make bt-mz CLASS=W NPROCS=4
**************************************************************
Building an NPB-MZ-MPI benchmark

- Specify the benchmark configuration
  - benchmark name: **bt-mz**, **lu-mz**, **sp-mz**
  - the number of MPI processes: **NPROCS=4**
  - the benchmark class (S, W, A, B, C, D, E): **CLASS=W**

```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
mpif77 -c -O3 -fopenmp bt.f  
[...]
cd ../common; mpif77 -c -O3 -fopenmp timers.f
mpif77 -O3 -fopenmp -o ../bin/bt-mz_W.4 \
bt.o initialize.o exact_solution.o exact_rhs.o set_constants.o \
adi.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/bt-mz_W.4
make: Leaving directory 'BT-MZ'
```
NPB-MZ-MPI / BT reference execution

- Launch as a hybrid MPI+OpenMP application

```
% cd bin
% OMP_NUM_THREADS=4 mpiexec -np 4 ./bt-mz_W.4
```

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)

<table>
<thead>
<tr>
<th>Time step</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time step</td>
<td>20</td>
</tr>
<tr>
<td>Time step</td>
<td>40</td>
</tr>
<tr>
<td>[...]</td>
<td></td>
</tr>
<tr>
<td>Time step</td>
<td>160</td>
</tr>
<tr>
<td>Time step</td>
<td>180</td>
</tr>
<tr>
<td>Time step</td>
<td>200</td>
</tr>
</tbody>
</table>

Verification Successful

BT-MZ Benchmark Completed.
Time in seconds = 5.57

Hint: save the benchmark output (or note the run time) to be able to refer to it later
Profile NPB-MZ-MPI / BT
NPB-MZ-MPI / BT Instrumentation

- 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 = mpif77
#
# Alternative variants to perform instrumentation
...
MPIF77 = scorep mpif77
#
# 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
  ```
  % make clean
  ```

- Re-build executable using Score-P compiler wrapper
  ```
  % 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 -lm
  ../sys/setparams bt-mz 4 B
  scorep mpif77 -c -O3 -fopenmp bt.f
  [...] 
  cd ../common;  scorep mpif77 -c -O3 -fopenmp timers.f 
  scorep mpif77 -O3 -fopenmp -o ../bin.scorep/bt-mz_W.4 
  bt.o initialize.o exact_solution.o exact_rhs.o set_constants.o 
  adi.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'
  ```
Measurement Configuration: scorep-info

- Score-P measurements are configured via environmental variables:

```bash
% 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 ...]
```
Summary Measurement Collection

- Change to the directory containing the new executable before running it with the desired configuration

```bash
% cd bin.scorep
% export OMP_NUM_THREADS=4
% export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt-mz_W_4x4_sum
% mpiexec -np 4 ./bt-mz_W_4
```

NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark

- Number of zones: 8 x 8
- Iterations: 200 dt: 0.000300
- Number of active processes: 4

Use the default load factors with threads

- Total number of threads: 16 (4.0 threads/process)

Calculated speedup = 15.96

Time step 1

[...] More application output [...]
BT-MZ Summary Analysis Report Examination

- 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
bt-mz_B.4  scorep_bt-mz_W_4x4_sum
% ls scorep_bt-mz_W_4x4_sum
profile.cubex  scorep.cfg
```
Congratulations!? 

- If you made it this far, you successfully used Score-P to 
  - instrument the application 
  - analyze its execution with a summary measurement, and 
  - examine it with one the interactive analysis report explorer GUIs 
- ... revealing the call-path profile annotated with 
  - the “Time” metric 
  - Visit counts 
  - MPI message statistics (bytes sent/received) 
- ... but how good was the measurement? 
  - The measured execution produced the desired valid result 
  - however, the execution took longer than expected! 
    - even when ignoring measurement start-up/completion 
    - it was probably dilated by instrumentation/measurement overhead
BT-MZ Summary Analysis Result Scoring

- Report scoring as textual output

```bash
% 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>9046029930</td>
<td>799.89</td>
<td>100.0</td>
</tr>
<tr>
<td>USR</td>
<td>9025830154</td>
<td>383.72</td>
<td>48.0 USR</td>
</tr>
<tr>
<td>OMP</td>
<td>19113728</td>
<td>411.49</td>
<td>51.4 OMP</td>
</tr>
<tr>
<td>COM</td>
<td>997150</td>
<td>0.75</td>
<td>0.1 COM</td>
</tr>
<tr>
<td>MPI</td>
<td>88898</td>
<td>3.92</td>
<td>0.5 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)

33.5 GB total memory
8.4 GB per rank!
## BT-MZ Summary Analysis Report Breakdown

- Score report breakdown by region

```
% scorep-score -r scorep_bt-mz_W_4x4_sum/profile.cubex

flt  type  max_tbc  time  %  region
ALL   9046029930  799.89  100.0  ALL
USR   9025830154  383.72  48.0  USR
OMP   19113728    411.49  51.4  OMP
COM   997150      0.75    0.1  COM
MPI   88898       3.92    0.5  MPI

USR   2894950740  152.50  19.1  binvcrhs_
USR   2894950740  98.73   12.3  matvec_sub_
USR   2894950740  117.78  14.7  matmul_sub_
USR   127716204   5.01    0.6  binvrhs_
USR   127716204   6.62    0.8  lhsinit_
USR   94933520    3.07    0.4  exact_solution_
OMP   1183488     0.04    0.0  !$omp parallel @exch_
OMP   1183488     0.04    0.0  !$omp parallel @exch_
OMP   1183488     0.04    0.0  !$omp parallel @exch_
```

More than 8 GB just for these 6 regions
**BT-MZ Summary Analysis Score**

- Summary measurement analysis score reveals
  - Total size of event trace would be ~34 GB
  - Maximum trace buffer size would be ~8.5 GB per rank
    - smaller buffer would require flushes to disk during measurement resulting in substantial perturbation
  - 99.8% of the trace requirements are for USR regions
    - purely computational routines never found on COM call-paths common to communication routines or OpenMP parallel regions
  - These USR regions contribute around 32% of total time
    - however, much of that is very likely to be measurement overhead for frequently-executed small routines

- Advisable to tune measurement configuration
  - Specify an adequate trace buffer size
  - Specify a filter file listing (USR) regions not to be measured
BT-MZ Summary Analysis Report Filtering

- Report scoring with prospective filter listing
  - 6 USR regions

```bash
% 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: 80814262 bytes
Estimated requirements for largest trace buffer (max_tbc): 20203582 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.)

77 MB of memory in total, 20 MB per rank!
BT-MZ Summary Analysis Report Filtering

- Score report breakdown by region

```bash
% scorep-score -r -f ../config/scorep.filt \
> scorep_bt-mz_W_4x4_sum/profile.cubex
```

<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>20203582</td>
<td>416.17</td>
<td>52.0 ALL-FLT</td>
</tr>
<tr>
<td>+ FLT</td>
<td>9025826370</td>
<td>383.72</td>
<td>48.0 FLT</td>
</tr>
<tr>
<td>- OMP</td>
<td>19113728</td>
<td>411.49</td>
<td>51.4 OMP-FLT</td>
</tr>
<tr>
<td>* COM</td>
<td>997150</td>
<td>0.75</td>
<td>0.1 COM-FLT</td>
</tr>
<tr>
<td>- MPI</td>
<td>88898</td>
<td>3.92</td>
<td>0.5 MPI-FLT</td>
</tr>
<tr>
<td>* USR</td>
<td>3806</td>
<td>0.00</td>
<td>0.0 USR-FLT</td>
</tr>
<tr>
<td>+ USR</td>
<td>2894950740</td>
<td>152.50</td>
<td>19.1 binvcrhs_</td>
</tr>
<tr>
<td>+ USR</td>
<td>2894950740</td>
<td>98.73</td>
<td>12.3 matvec_sub_</td>
</tr>
<tr>
<td>+ USR</td>
<td>2894950740</td>
<td>117.78</td>
<td>14.7 matmul_sub_</td>
</tr>
<tr>
<td>+ USR</td>
<td>127716204</td>
<td>5.01</td>
<td>0.6 binvrhs_</td>
</tr>
<tr>
<td>+ USR</td>
<td>127716204</td>
<td>6.62</td>
<td>0.8 lhsinit_</td>
</tr>
<tr>
<td>+ USR</td>
<td>94933520</td>
<td>3.07</td>
<td>0.4 exact_solution_</td>
</tr>
<tr>
<td>- OMP</td>
<td>1183488</td>
<td>0.04</td>
<td>0.0 !$omp parallel @exch_...</td>
</tr>
<tr>
<td>- OMP</td>
<td>1183488</td>
<td>0.04</td>
<td>0.0 !$omp parallel @exch_...</td>
</tr>
<tr>
<td>- OMP</td>
<td>1183488</td>
<td>0.04</td>
<td>0.0 !$omp parallel @exch_...</td>
</tr>
</tbody>
</table>

Filtered routines marked with ‘+’
BT-MZ Filtered Summary Measurement

- Set new experiment directory and re-run measurement with new filter configuration
  - Adjust configuration and re-run measurement

- Submit job

```bash
% export OMP_NUM_THREADS=4
% export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt-mz_W_4x4_sum_with_filter
% export SCOREP_FILTERING_FILE=../config/scorep.filt
% mpiexec -np 4 ./bt-mz_W.4
```

NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark

Number of zones: 8 x 8  
Iterations: 200  
dt: 0.000300  
Number of active processes: 4

Use the default load factors with threads  
Total number of threads: 16 (4.0 threads/process)

Calculated speedup = 15.96

Time step 1

[... More application output ...]
BT-MZ Tuned Summary Analysis Report Score

- Scoring of new analysis report as textual output

```
% scorep-score scorep_bt-mz_W_4x4_sum_with_filter/profile.cubex
Estimated aggregate size of event trace: 80814262 bytes
Estimated requirements for largest trace buffer (max_tbc): 20203582 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.)
```

<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>20203582</td>
<td>218.95</td>
<td>100.0</td>
</tr>
<tr>
<td>OMP</td>
<td>19113728</td>
<td>216.94</td>
<td>99.1</td>
</tr>
<tr>
<td>COM</td>
<td>997150</td>
<td>0.73</td>
<td>0.3</td>
</tr>
<tr>
<td>MPI</td>
<td>88898</td>
<td>1.27</td>
<td>0.6</td>
</tr>
<tr>
<td>USR</td>
<td>3806</td>
<td>0.00</td>
<td>0.0</td>
</tr>
</tbody>
</table>

- Significant reduction in runtime (measurement overhead)
  - Not only reduced time for USR regions, but MPI/OMP reduced too!

- Further measurement tuning (filtering) may be appropriate
  - E.g., use “timer_*” to filter timer_start_, timer_read_, etc.
Advanced Measurement Configuration: Metrics

- Recording hardware counters via PAPI

  ```
  % export SCOREP_METRIC_PAPI=PAPI_L2_TCM,PAPI_FP_OPS
  ```

- Also possible to record them only per rank

  ```
  % export SCOREP_METRIC_PAPI_PER_PROCESS=PAPI_L3_TCM
  ```

- Recording operating system resource usage

  ```
  % export SCOREP_METRIC_RUSAGE_PER_PROCESS=ru_maxrss,ru_stime
  ```
Advanced Measurement Configuration: Metrics

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

Note:
Due to hardware restrictions
- number of concurrently recorded events is limited
- there may be invalid combinations of concurrently recorded events
Advanced Measurement Configuration: Metrics

- Available resource usage metrics

```
% man getrusage
[... Output ...]

struct rusage {
    struct timeval ru_utime; /* user CPU time used */
    struct timeval ru_stime; /* system CPU time used */
    long ru_maxrss;    /* maximum resident set size */
    long ru_ixrss;    /* integral shared memory size */
    long ru_idrss;    /* integral unshared data size */
    long ru_isrss;    /* integral unshared stack size */
    long ru_minflt;  /* page reclaims (soft page faults) */
    long ru_majflt;  /* page faults (hard page faults) */
    long ru_nswap;  /* swaps */
    long ru_inblock; /* block input operations */
    long ru_oublock; /* block output operations */
    long ru_msgsnd; /* IPC messages sent */
    long ru_msgrcv; /* IPC messages received */
    long ru_nsignals; /* signals received */
    long ru_nvcsw; /* voluntary context switches */
    long ru_nivcsw; /* involuntary context switches */
};
[... More output ...]
```

Note:
(1) Not all fields are maintained on each platform.
(2) Check scope of metrics (per process vs. per thread)
BT-MZ Trace Measurement Collection...

- Adjust configuration and re-run the application using the tracing mode of Score-P

```bash
% export OMP_NUM_THREADS=4
% export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt-mz_W_4x4_trace
% export SCOREP_FILTERING_FILE=../config/scorep.filt
% export SCOREP_ENABLE_TRACING=true
% export SCOREP_ENABLE_PROFILING=false
% export SCOREP_TOTAL_MEMORY=30M
% mpiexec -np 4 ./bt-mz_W

NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark

[... More application output ...]
```

- Separate trace file per thread written straight into new experiment directory
  .:/scorep_bt-mz_B_4x4_trace
Visualize NPB-MZ-MPI/BT with Vampir
Start Vampir

% vampir <tracefile>

% vampir scorep_bt-mz_W_4x4_trace/traces.otf2

- Start Vampir and load trace
Visualization of the NPB-MZ-MPI / BT trace