



TECHNISCHE
UNIVERSITÄT
DRESDEN

Center for Information Services and High Performance Computing (ZIH)

User Library Wrapping for Score-P

Ronny Brendel, Bert Wesarg, Sebastian Öste (TU Dresden)

15th Dec 2016, Ostrava, IT4I

Performance Optimization and Productivity Tutorial



Contents

- Motivation
- What Does This Gromacs Run Do?
- How To Trace It
 - Score-P Installation
 - Create FFTW Wrapper
 - Build and Execute Gromacs
- The Trace
- Non-admin / User Perspective
- Conclusion & Future Work

Motivation

- Score-P works only with relinking (and most of the time requires recompiling) the whole application
 - User Library Wrapping is one step towards analyzing unmodified executables
- You cannot see calls to libraries (without manual instrumentation)
- You could lower measurement overhead and tracefile size
 - `scorep-score` filtering workflow could be avoided

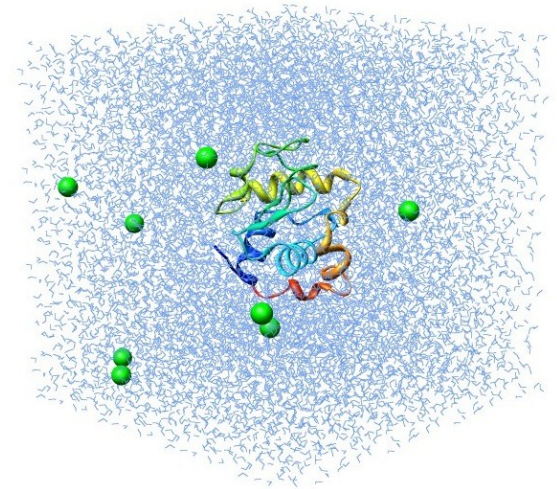


What Does This Gromacs Run Do

- Tutorial: **Lysozyme in Water** (Author: Justin A. Lemkul, Ph.D.)

<http://www.bevanlab.biochem.vt.edu/Pages/Personal/justin/gmx-tutorials/lysozyme/index.html>

- Lyszyme: Enzymes that damage bacterial cell walls
- Give a protein into a box of water, add ions and let it react
 - Define Molecules (Protein and Solvent)
 - Define Box and Solvate
 - Add Ions
 - Prepare Molecular Dynamics Simulation
 - Energy Minimization
 - Equilibration
 - Calculate Molecular Dynamics
 - Analysis



How To Trace It > Score-P Installation

- Enable library wrapping via `--with-llvm=$LLVM_ROOT` during the configure step
- Unsure if it is enabled?
 - `$ scorep-info config-summary`
 - Search for „libwrap“

How To Trace It > Create FFTW Wrapper

- Execute `$ scorep-libwrap-init` for directions
- Tell `scorep-libwrap-init` how you would compile and link an application using FFTW

```
$ scorep-libwrap-init \
  --name=fftw \
  --prefix=$PREFIX \
  -x c \
  --cppflags="-O3 -DNDEBUG -openmp -I$FFTW_INC" \
  --ldflags="-L$FFTW_LIB" \
  --libs="-lfftw3f -lfftw3" \
  working_directory
```

- `$ cd working_directory`
- `$ ls` # (Check README.md for instructions)
- `$ make` # Generate and build wrapper
- `$ make check` # See if header analysis matches symbols
- `$ make install` #
- `$ make installcheck` # More checks: Linking etc.

How To Trace It > Build Gromacs

- Build Gromacs twice, once normal, once with Score-P and libwrap

- `$ module load intel fftw papi bullxmpi #Taurus TU-Dresden`

```
$ mkdir BUILD && cd BUILD
```

```
$ SCOREP_WRAPPER=off \
  cmake \
  -DCMAKE_CXX_COMPILER=scorep-icpc \
  -DCMAKE_C_COMPILER=scorep-icc \
  -DCMAKE_INSTALL_PREFIX=$PREFIX \
  -DGMX_MPI=on \
  -DGMX_GPU=off \
  -DFFTW_INCLUDE_DIR=$FFTW_INC \
  -DFFTW_LIBRARY=$FFTW_LIB/libfftw3f.so \
  ..
```

```
$ make \
  SCOREP_WRAPPER_INSTRUMENTER_FLAGS="--  
libwrap=runtime:fftw --thread=omp --mutex=omp  
--nocompiler" \
  SCOREP_WRAPPER_COMPILER_FLAGS="-D_GNU_SOURCE"
```

How To Trace It > Execute Gromacs

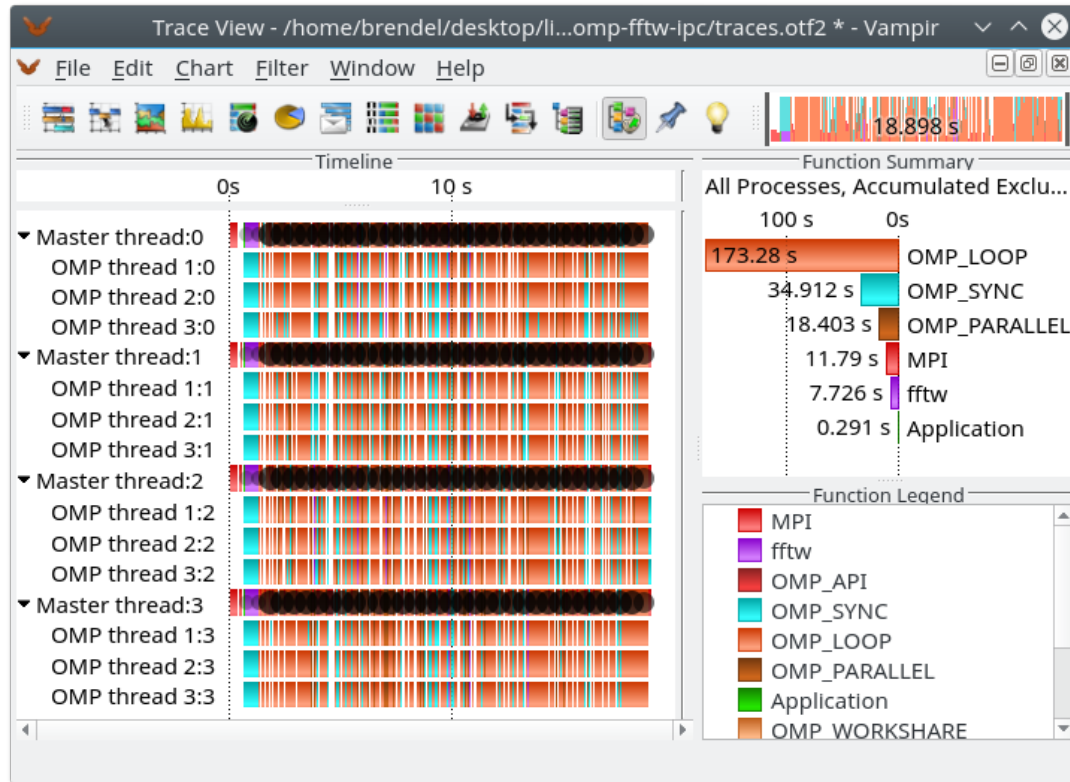
- This Gromacs run consists of 14 steps
- Run all of them normal and only the final molecular dynamics run with the instrumented installation

```
SCOREP_ENABLE_TRACING=yes
```

```
SCOREP_TOTAL_MEMORY=300M
```

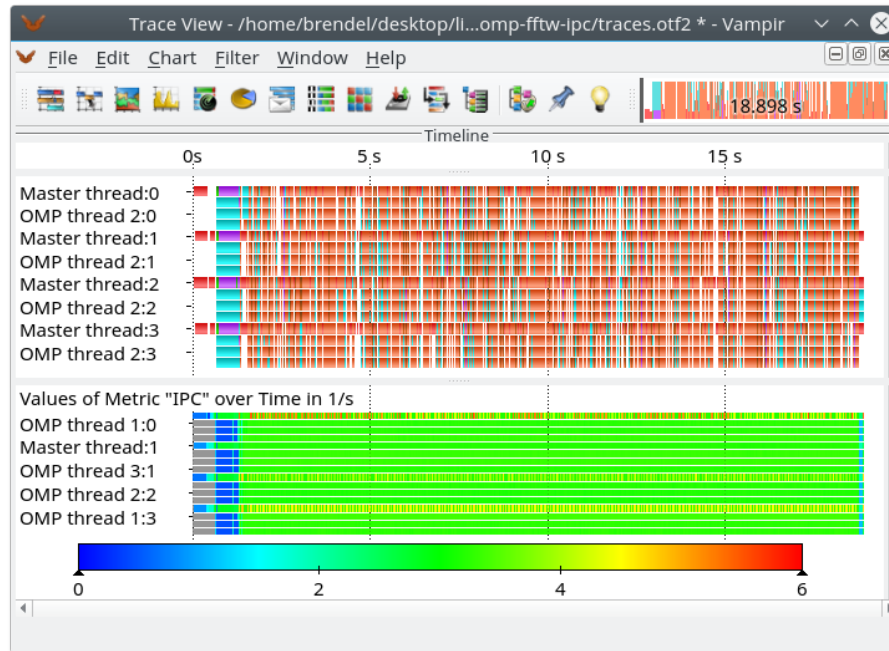
```
SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC # for IPC
```


The Trace > Overview

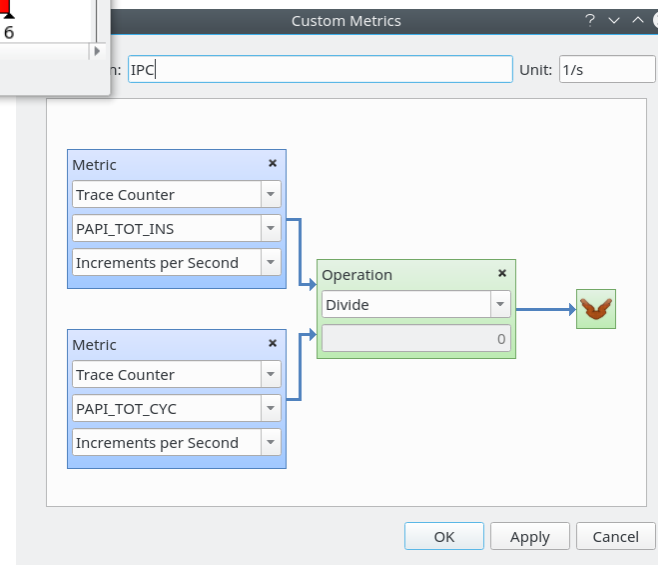


- MPI+OpenMP
- Some sync
- FFTW!
- Simple to trace!
- Low overhead!
- Only 300MB
- Full trace >300GB
- No filtering necessary since only threading, MPI and FFTW has been traced!

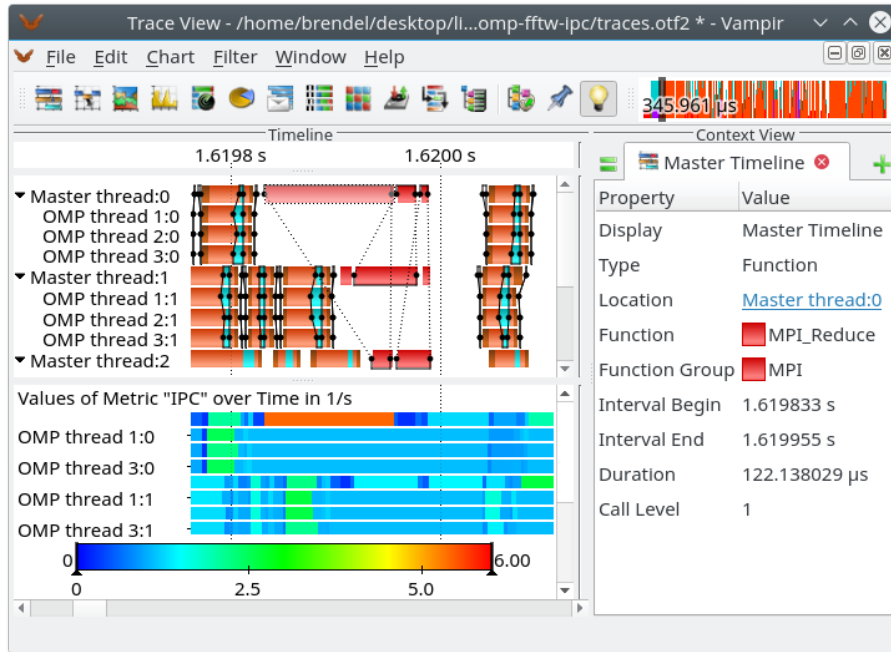
The Trace > Instructions Per Cycle



- Up to 6 max (which is about right)
- Mostly around 2-3
- Very high IPC on process 0
 - Why is that? Zoom in.



The Trace > Instructions Per Cycle

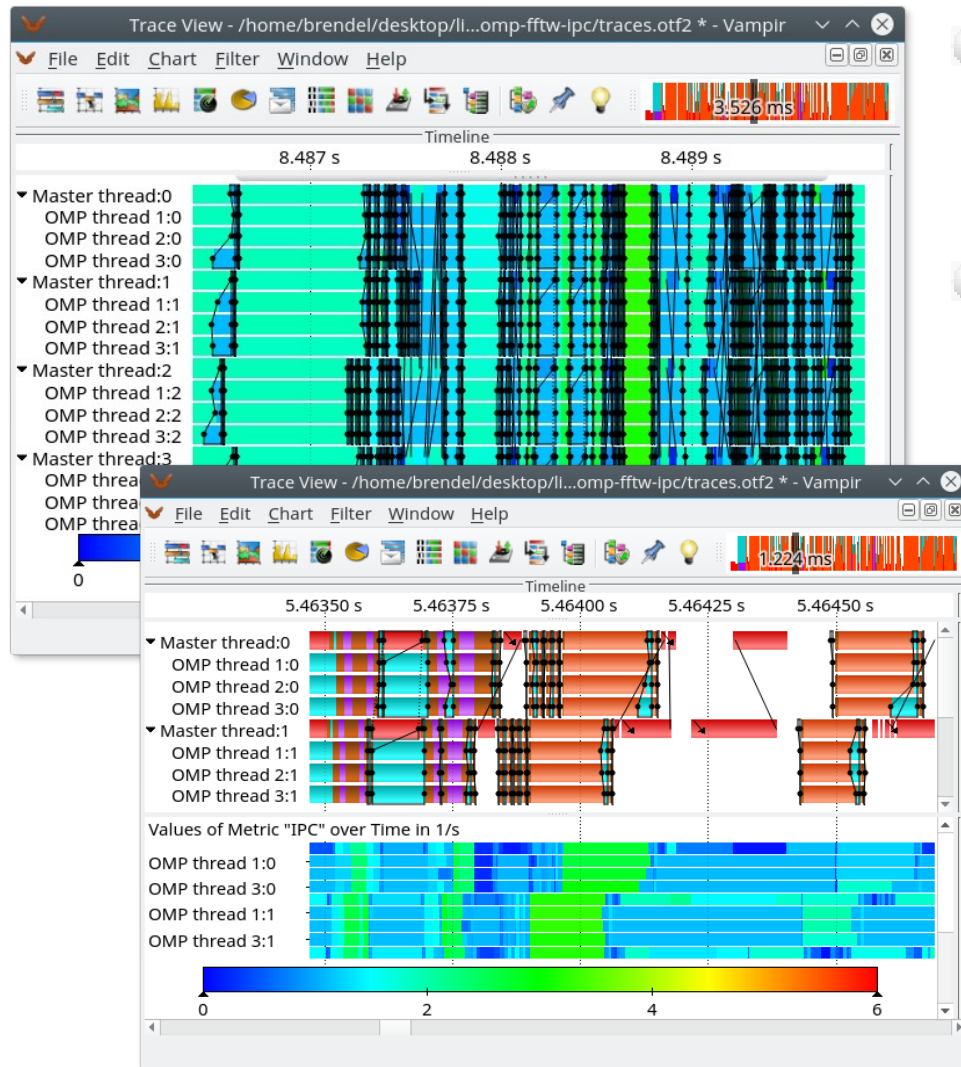


6 IPC in MPI_Reduce :(

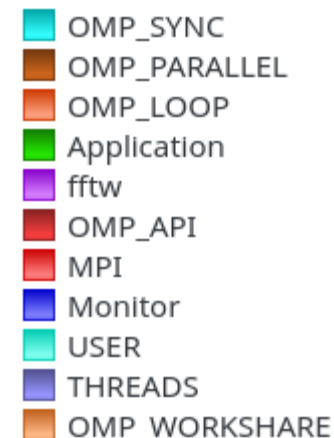
→ Busy waiting

→ High IPC doesn't always mean your code does well

The Trace > Instructions Per Cycle



- IPC during iterations
 - Overlay timeline with performance data or not
- Differing IPC → Visualize to gain insight into your code's efficiency



Non-admin / User Perspective

- Ideally, the administrators already installed wrappers for certain libraries
- Use `scorep --help` or
`scorep-info libwrap-summary [--full] [wrappername]`
to query which wrappers are installed
- Link with:
 - `scorep --libwrap=[wrapmode:] [wrappername] $CC ...`

Conclusions

- Simple to wrap a C/C++ library
 - Works even if you don't have the source
- Simple to use
- Smaller trace, lower overhead
 - No filtering / `scorep-score` needed
- Calls to your library are visible
- Ideal for recording performance counters, since the granularity is more coarse than full trace runs and calls to the library are recorded

- Download:

<https://cloudstore.zih.tu-dresden.de/index.php/s/2FBnJbUwp5LjcBe>

- Since events from unknown threads are not supported, in general it is necessary to recompile and link the whole application
 - Remove this requirement, since in principle you could use library wrapping without modifying the executable
- Improve C++ handling
 - C++ can be problematic because LLVM doesn't provide everything we need at the moment

Thank you!