VAMPIR & VAMPIRTRACE Hands On

8th VI-HPS Tuning Workshop at RWTH Aachen
September, 2011

Tobias Hilbrich and Joachim Protze

Slides by: Andreas Knüpfer, Jens Doleschal,

ZIH, Technische Universität Dresden

Hands-on: NPB – Build

- Copy the tutorial directory (if necessary)

```
% cp -r ~hpclab01/tutorial/NPB3.3-MZ-MPI/ ./
```

- Move into tutorial directory in your home directory

```
% cd NPB3.3-MZ-MPI
```

- Select the VampirTrace compiler wrappers

```
% vim config/make.def
      -> comment out line 32, resulting in:
                  ...
                  33: #MPIF77 = mpif77
                  ...
      -> remove the comment from line 40, resulting in:
                  ...
                  40: MPIF77 = vtf77 –vt:hyb –vt:f77 mpif77
                  …
```

Hands-on: NPB – Run

- Set up modules

```
module switch openmpi intelmpi
module load UNITE \
       vampirtrace/5.11-intel2-intel-marmot-papi
```

Aachen (Cluster-beta)

```
module load UNITE vampirtrace
```

Juropa

- Build benchmark

```
% make clean; make suite
```

- Go to bin directory

```
% cd bin.vampir
```

Hands-on: NPB – Run

- Create and edit the jobscript

```
cp ../jobscript/run.lsf ./
vim run.lsf
```

```
cp ../jobscript/run.msub ./
vim run.msub
```

- Jobscript:

```
#!/usr/bin/env zsh
# submit this job with "bsub < run.lsf"

#BSUB -J mzmpibt
…
#BSUB -U vihps

export OMP_NUM_THREADS=6

module swap openmpi intelmpi
module load UNITE \
       vampirtrace/5.11-intel2-intel-marmot-papi
module list

set -x

$MPIEXEC $FLAGS_MPI_BATCH  bt-mz_B.4
```

```
#!/bin/bash
# submit with "msub run.msub"
#MSUB -N mzmpibt
…
#MSUB -j oe

cd $PBS_O_WORKDIR

# benchmark configuration
export OMP_NUM_THREADS=4
PROCS=4
CLASS=B
EXE=./bt-mz_$CLASS.$PROCS

module load UNITE vampirtrace

mpiexec -np $PROCS --envall $EXE
```

Hands-on: NPB – Run

- Submit the job

```
bsub < run.lsf
```

```
msub run.msub
```

- Investigate the output file

```
% cat mzmpibt.<ID>

NAS Parallel Benchmarks (NPB3.2-MZ-MPI) - BT-MZ MPI+OpenMP Benchmark

Number of zones:   4 x   4
Iterations: 200    dt:   0.000800
Number of active processes:     4

Use the default load factors with threads
Total number of threads:     16   (  4.0 threads/process)
Calculated speedup =     15.64
Time step    1
[0]VampirTrace: Maximum number of buffer flushes reached \
(VT_MAX_FLUSHES=1)
[0]VampirTrace: Tracing switched off permanently
…
```

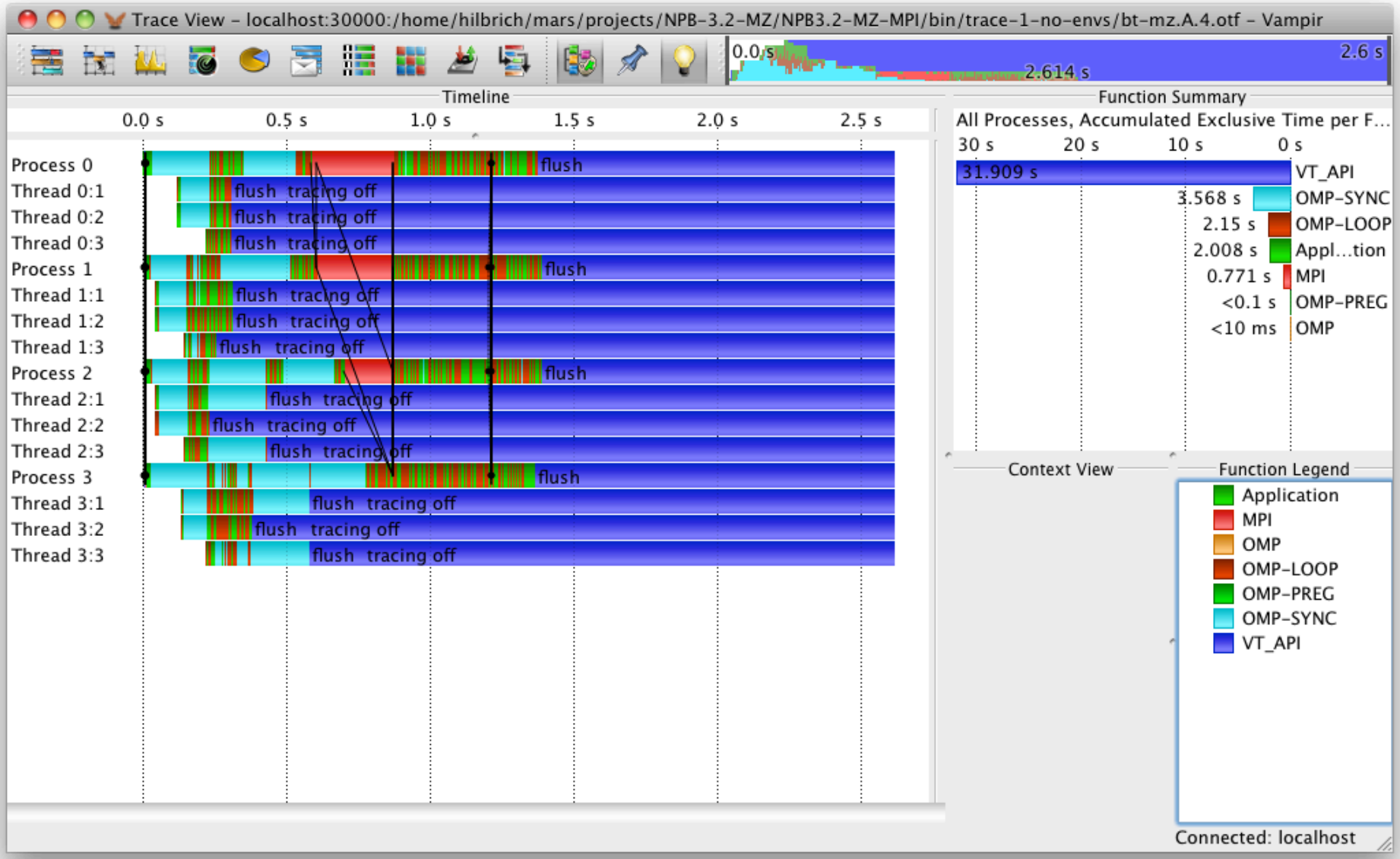
- Resulting trace files

```
% ls
bt-mz_B.4
bt-mz_B.4.0.def.z
bt-mz_B.4.1.events.z
bt-mz_B.4.10001.events.z
bt-mz_B.4.10002.events.z
...
bt-mz_B.4.40004.events.z
bt-mz_B.4.50001.events.z
bt-mz_B.4.50002.events.z
bt-mz_B.4.50003.events.z
bt-mz_B.4.50004.events.z
bt-mz_B.4.otf
```

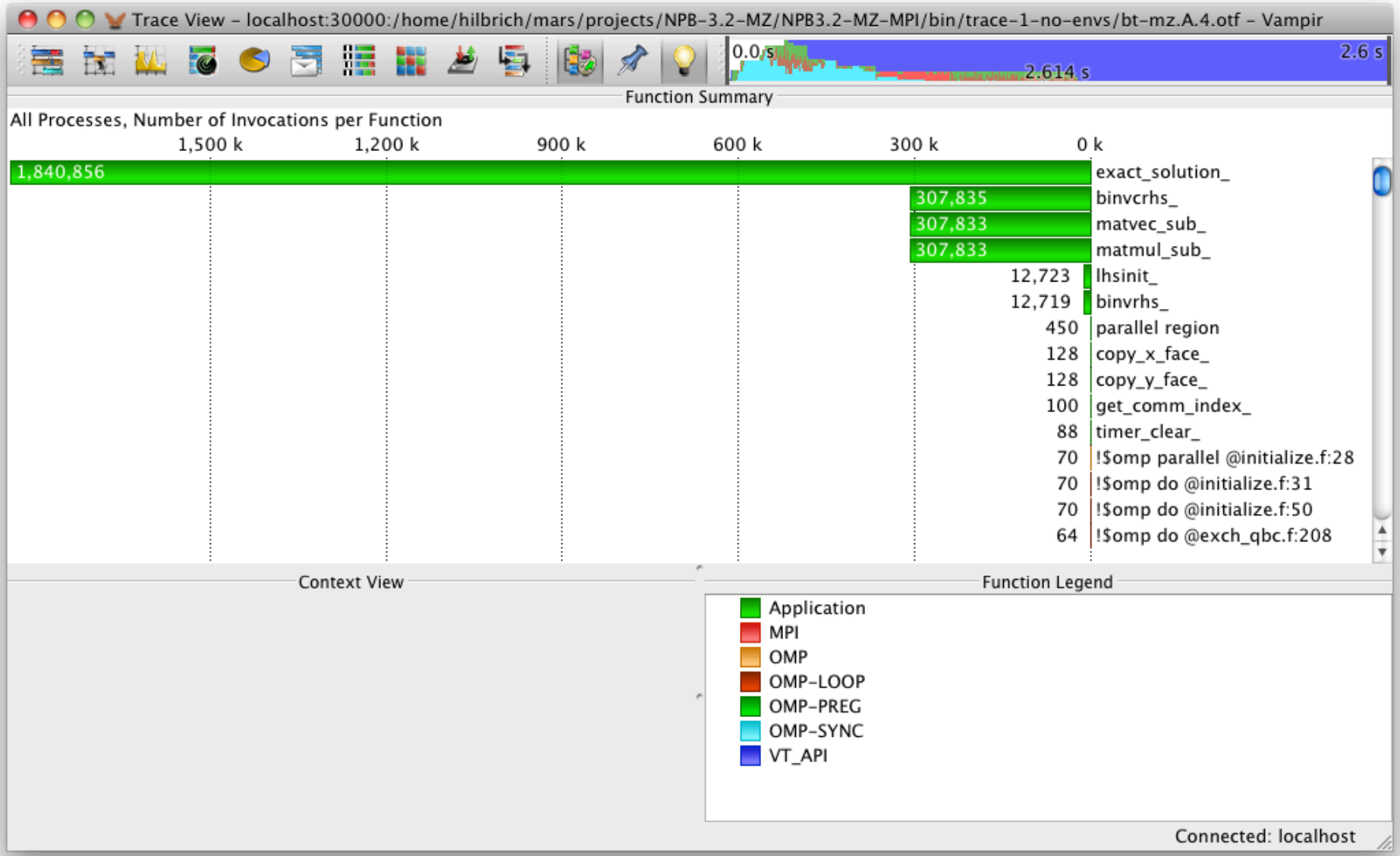
- Visualization with Vampir7

```
% module load vampir
% vampir bt-mz_B.4.otf
```

Hands-on: NPB – Initial Trace



Hands-on: NPB – Initial Trace



Hands-on: NPB – Buffer Size

- Decrease number of buffer flushes by increasing the buffer size

```
% export VT_BUFFER_SIZE=350M
```

- Set a new file prefix

Add both to your jobscript

```
% export VT_FILE_PREFIX=bt_2_buffer_350M
```

- Resubmit your jobscript

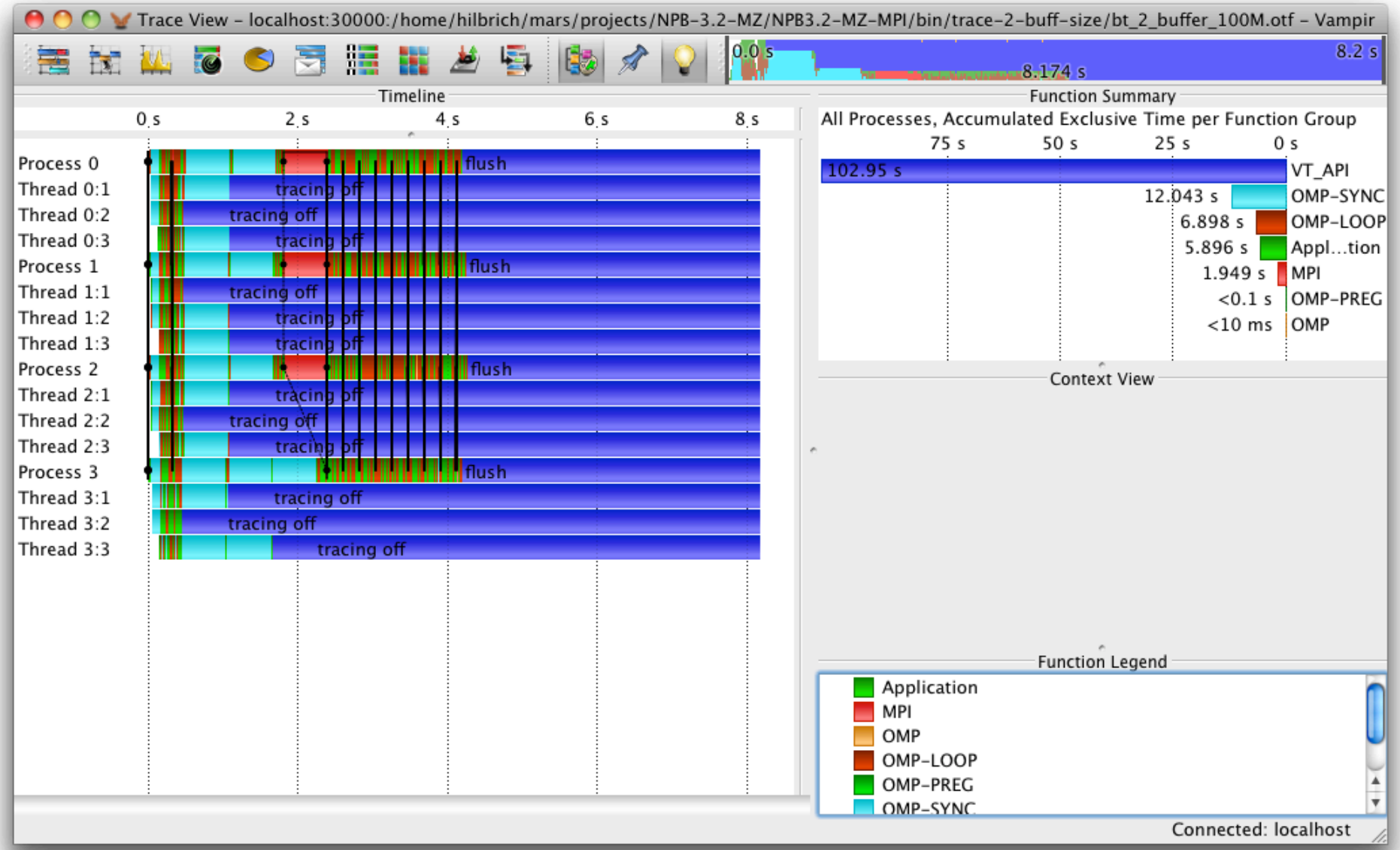
```
bsub < run.lsf
```

```
msub run.msub
```

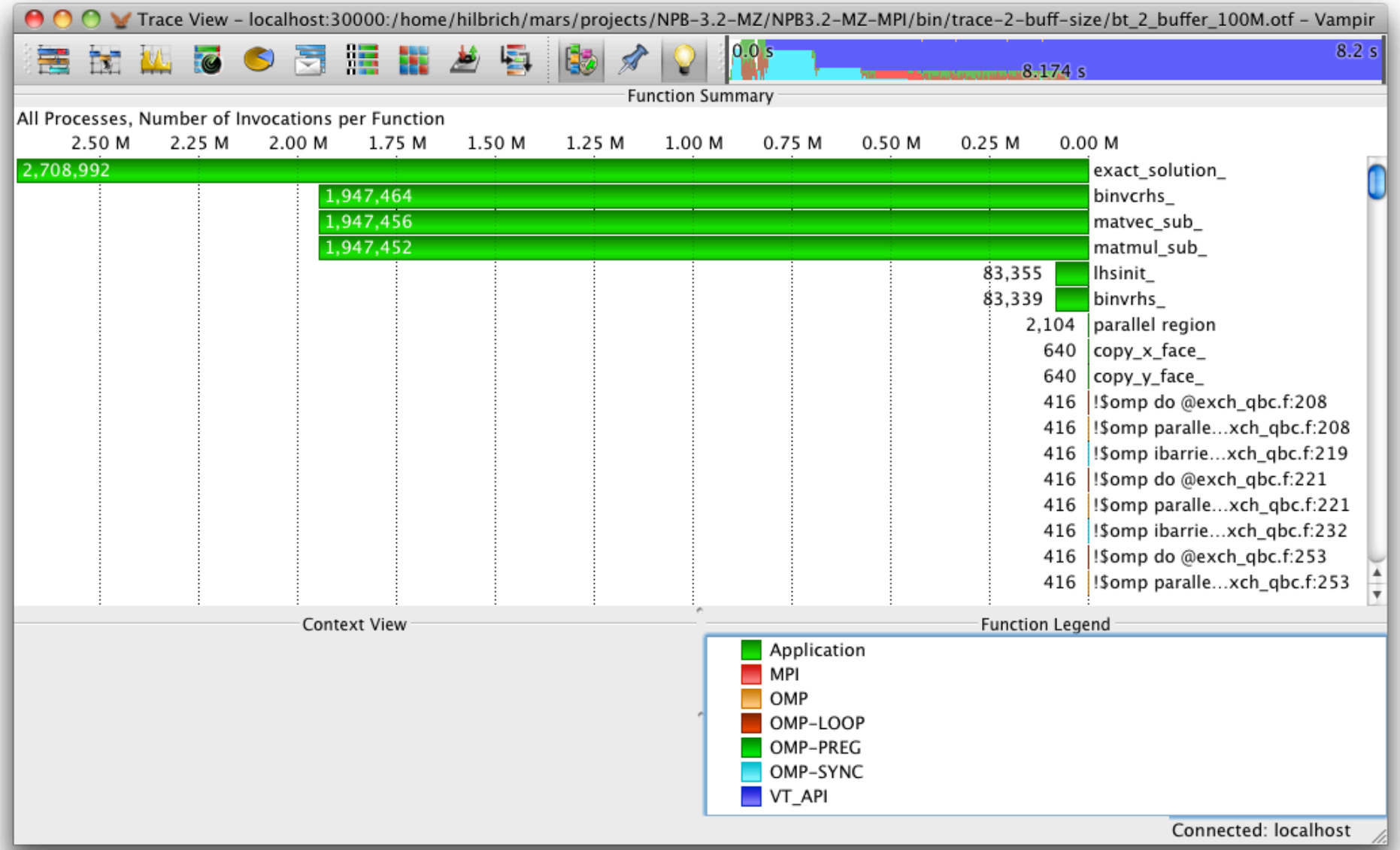
- Analyze the new trace

```
% vampir bt_2_buffer_350M.otf
```

Hands-on: NPB – Second Trace



Hands-on: NPB – Second Trace



Function Filtering

- Limit trace size with filtering
- Environment variable VT_FILTER_SPEC

Example, don't type it into the shell

```
% export VT_FILTER_SPEC = /home/user/filter.spec
```

- Filter definition file contains a list of filters

```
my_*;test_* -- 1000
debug_* -- 0
calculate -- -1
* -- 1000000
```

- See also the vtfilter tool
  - can generate a customized filter file
  - can reduce the size of existing trace files

Function Grouping

- Groups can be defined for related functions
  - Groups can be assigned different colors, highlighting different activities
- Environment variable VT_GROUPS_SPEC

```
% export VT_GROUPS_SPEC = /home/user/groups.spec
```

- Group file contains a list of associated entries

```
CALC=calculate
MISC=my*;test
UNKNOWN=*
```

Hands-on: NPB – Filtering

- Write a filter specification

```
% vim vt_filter.txt
      exact_solution_ -- 0
      binvcrhs_ -- 0
      matvec_sub_ -- 0
      matmul_sub_ -- 0
      lhsinit_ -- 0
      binvrhs_ -- 0
```

- Activate filtering and set a new file prefix

```
% export VT_FILTER_SPEC=vt_filter.txt
% export VT_FILE_PREFIX=bt_3_filter
```

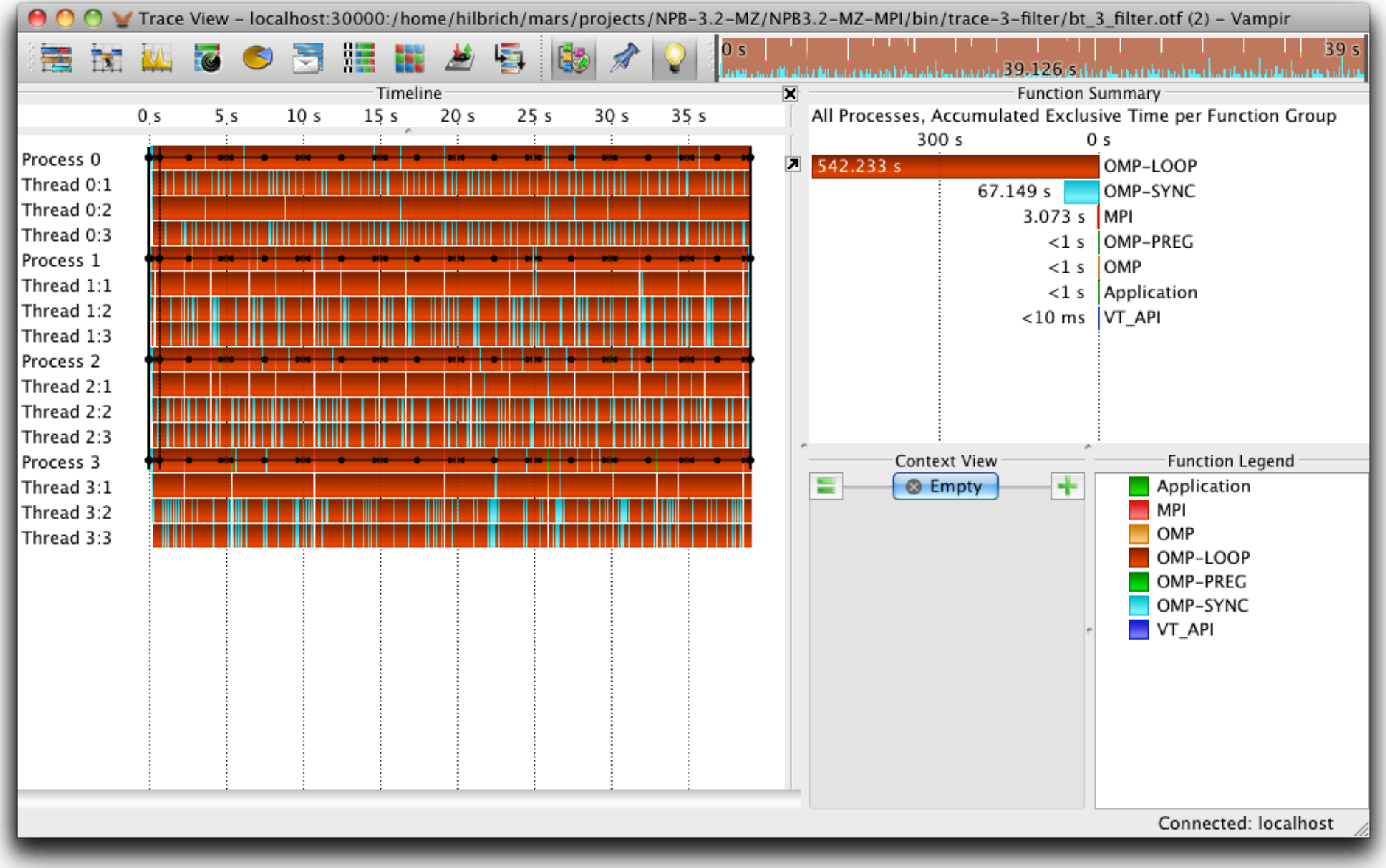
- Resubmit jobscript

Add both to your jobscript, keep "export VT_BUFFER_SIZE=350M"

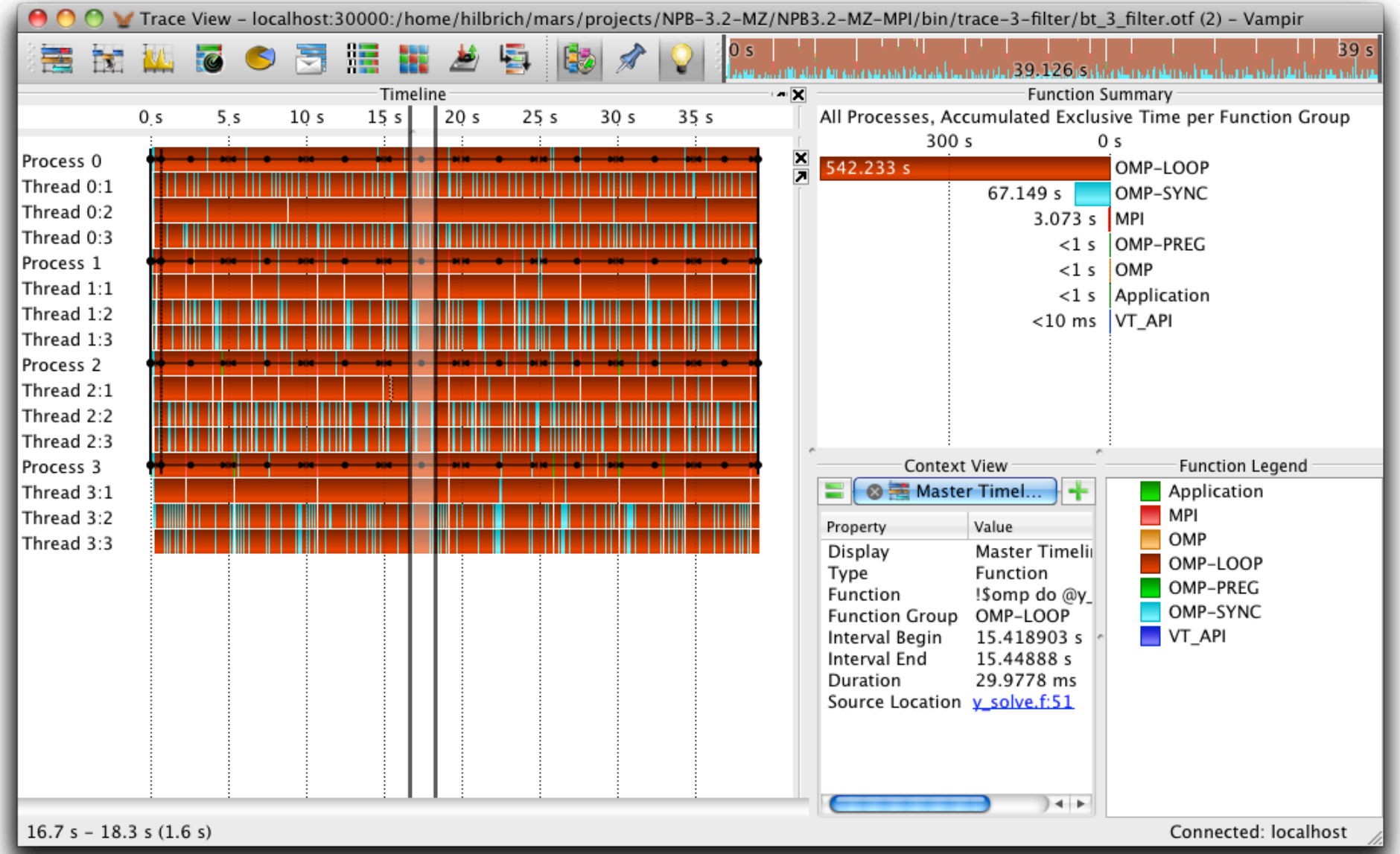
```
bsub < run.lsf
```

```
msub run.msub
```

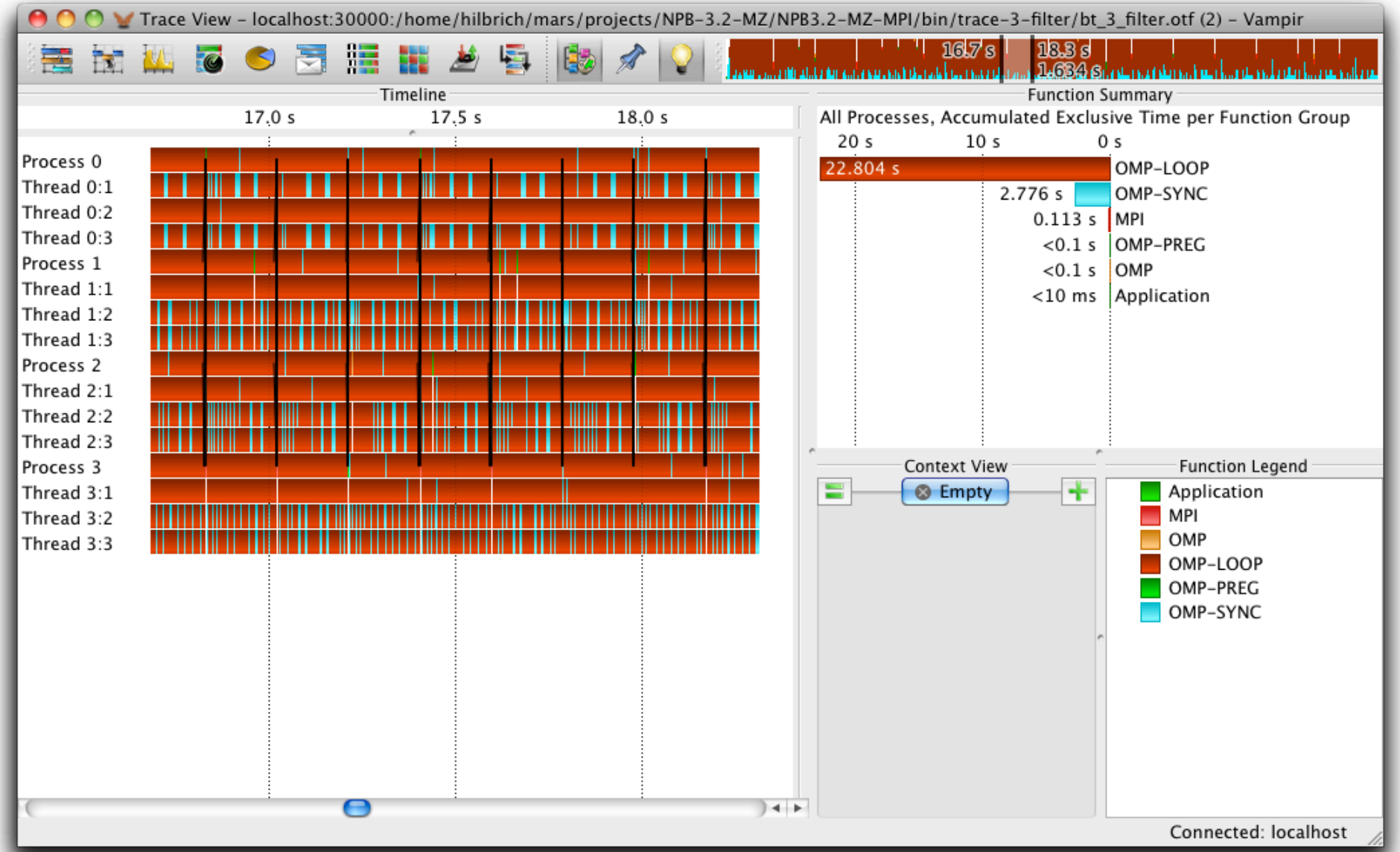
Hands-on: NPB – Filtered Trace



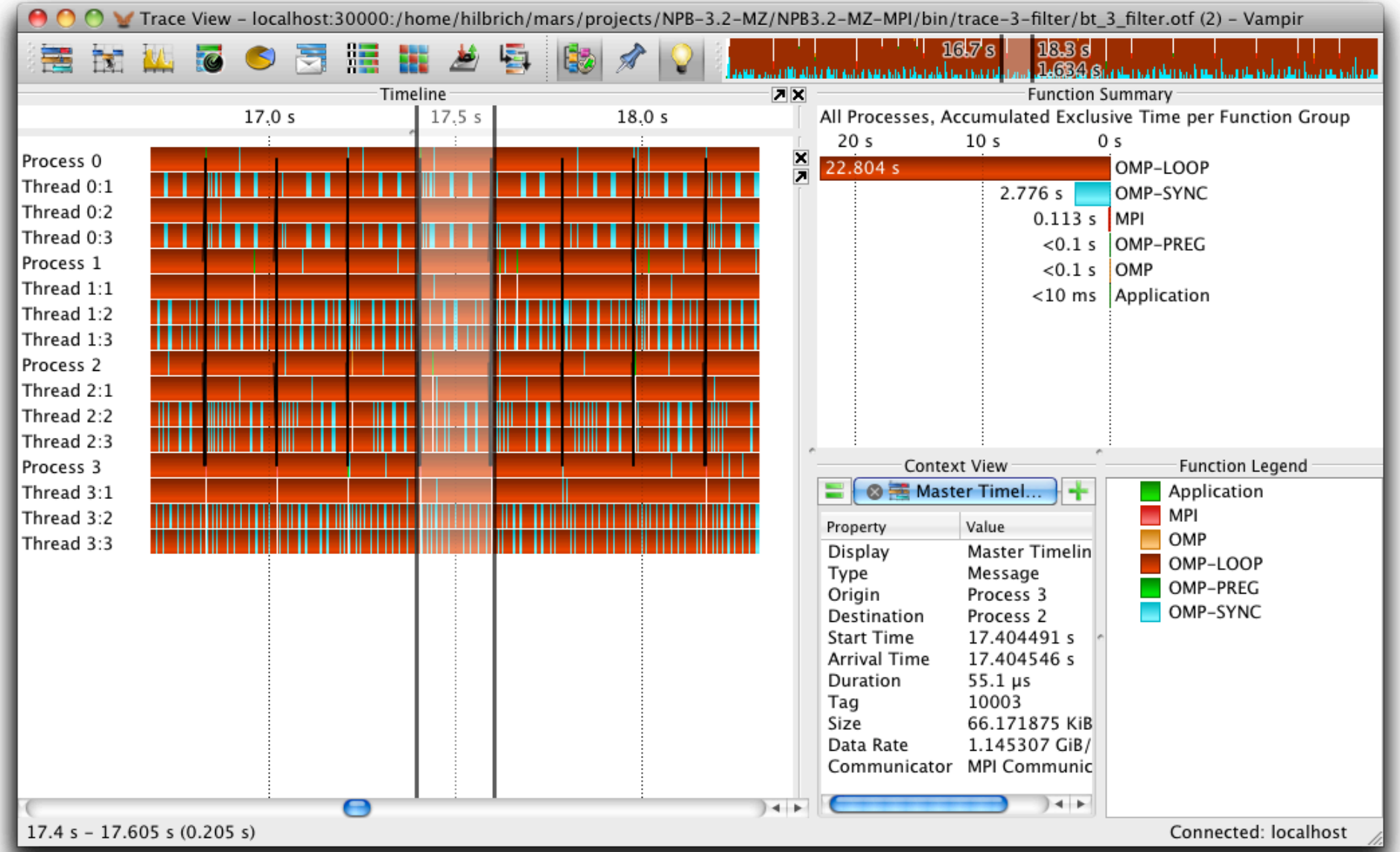
Hands-on: NPB – Filtered Trace



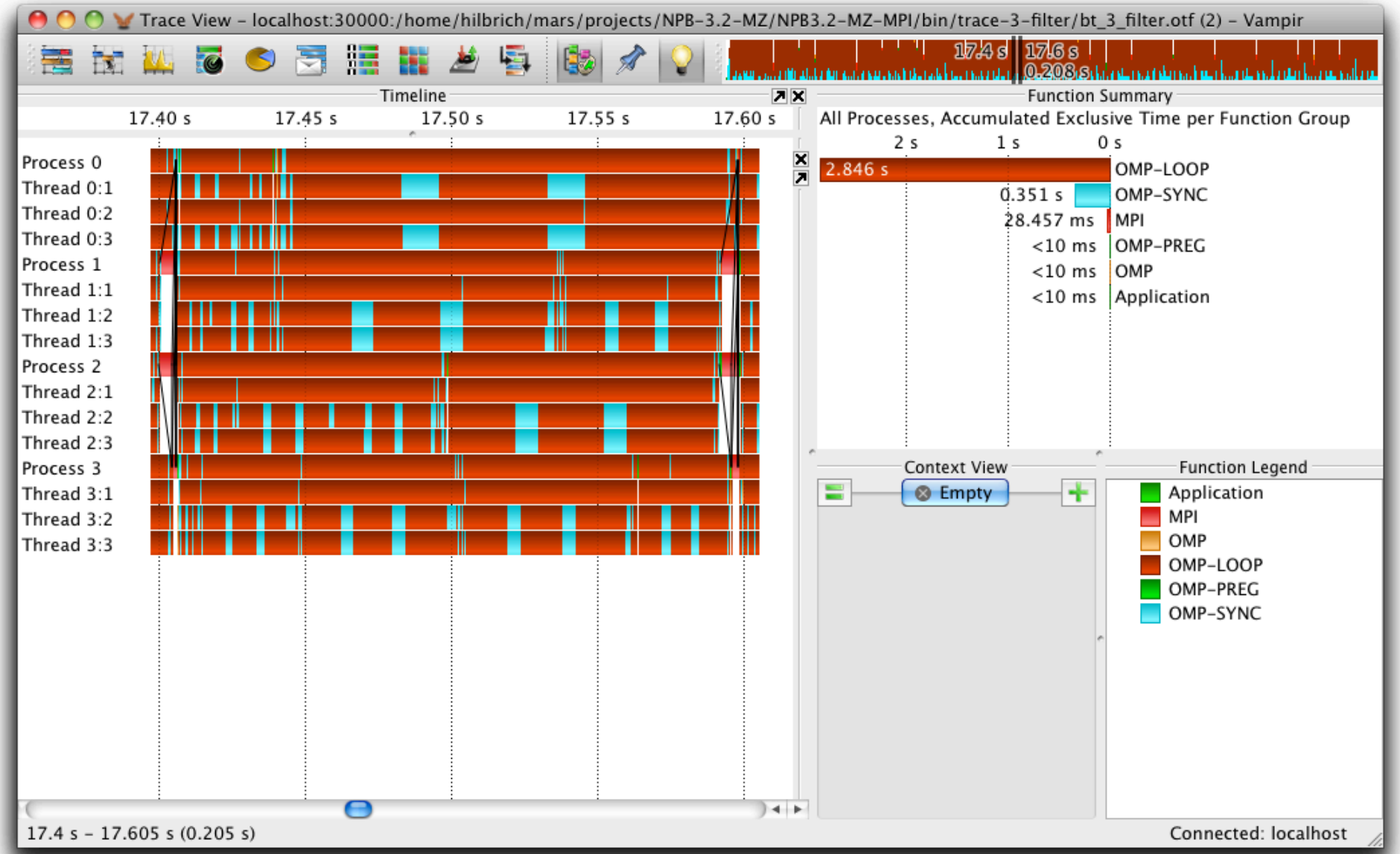
Hands-on: NPB – Filtered Trace



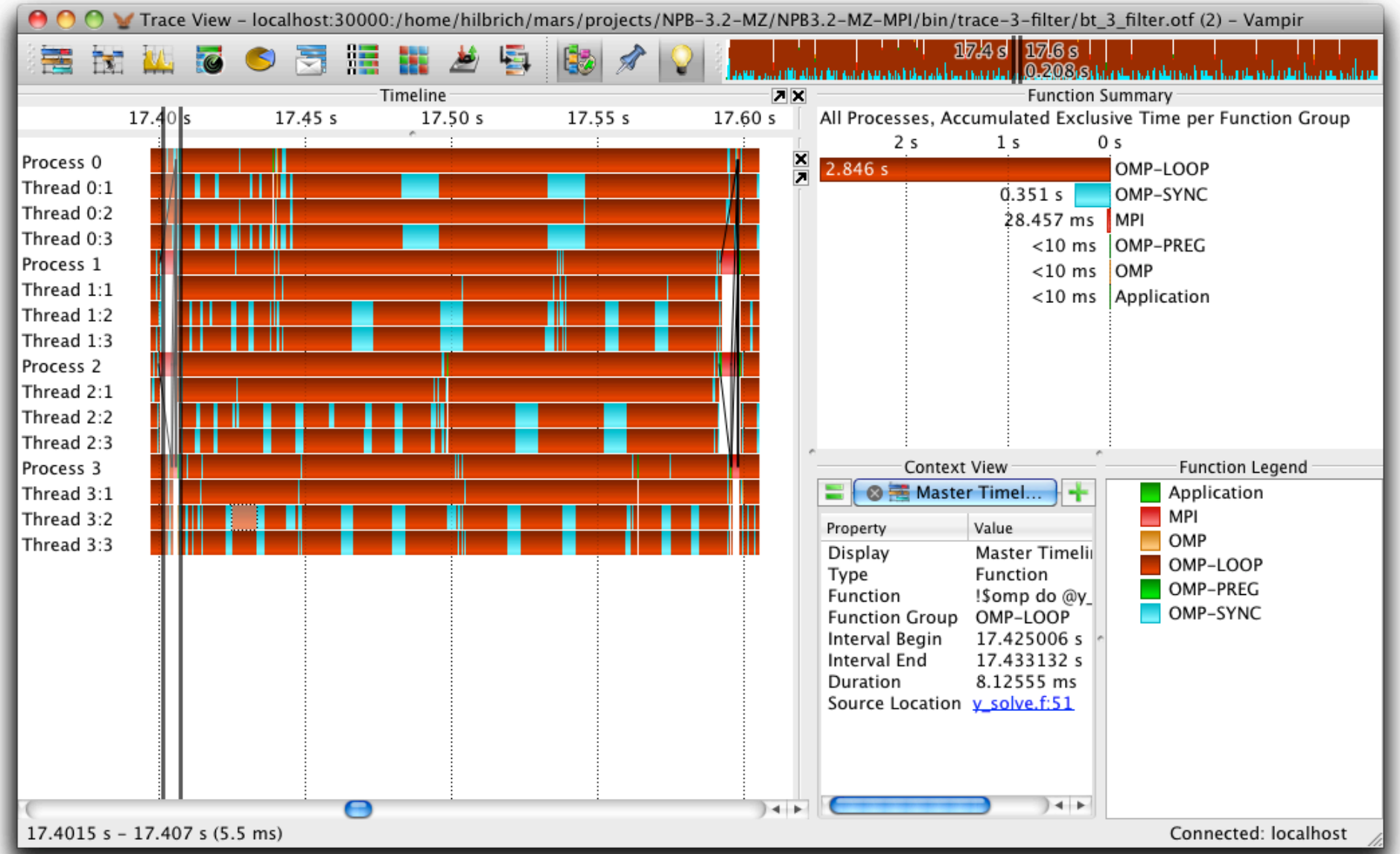
Hands-on: NPB – Filtered Trace



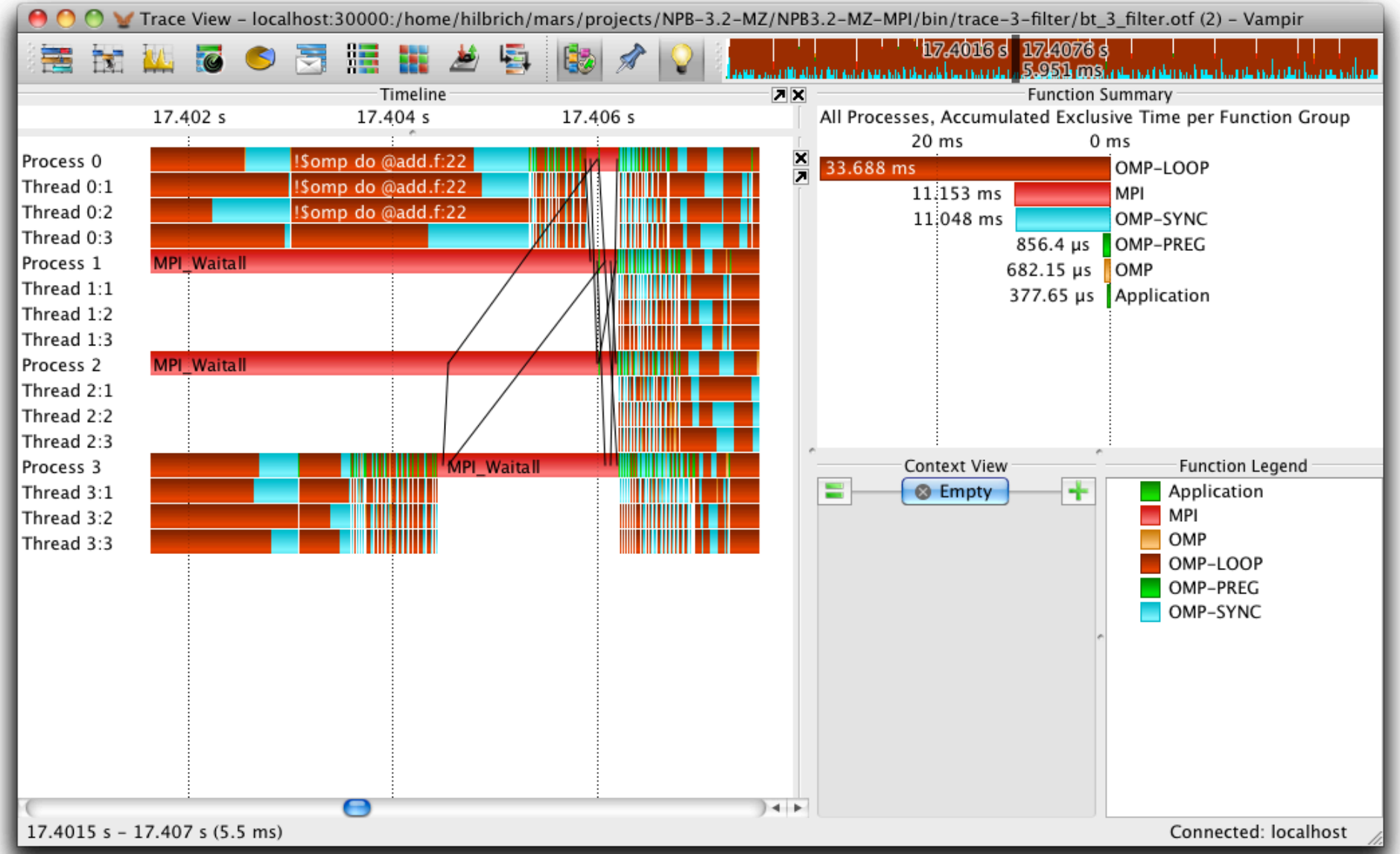
Hands-on: NPB – Filtered Trace



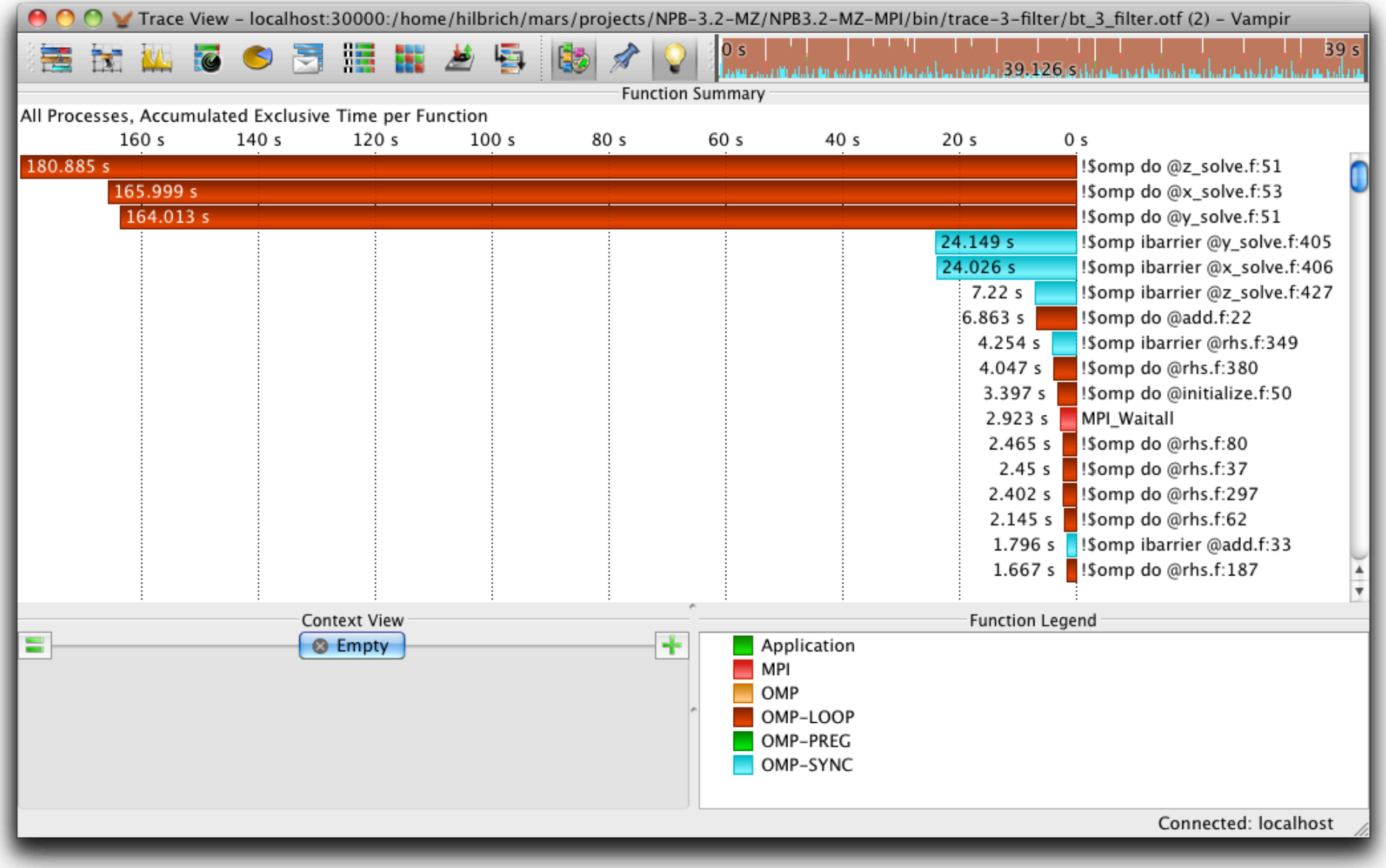
Hands-on: NPB – Filtered Trace



Hands-on: NPB – Filtered Trace



Hands-on: NPB – Filtered Trace



PAPI

- PAPI counters can be included in traces
  - If VampirTrace was build with PAPI support
  - If PAPI is available on the platform
- VT_METRICS specifies a list of PAPI counters

```
% export VT_METRICS = PAPI_FP_OPS:PAPI_L2_TCM
```

- see also the PAPI commands papi_avail and papi_command_line

Memory Allocation and I/O counters

- Memory allocation counters can be recorded:
  - If VampirTrace build with memory allocation tracing support
  - If GNU glibc is used on the platform
- intercept glibc functions like “malloc” and “free”
- Environment variable VT_MEMTRACE

```
% export VT_MEMTRACE = yes
```

- I/O counters can be included in traces
  - If VampirTrace was build with I/O tracing support
- Standard I/O calls like “open” and “read” are recorded
- Environment variable VT_IOTRACE

```
% export VT_IOTRACE = yes
```

- Record PAPI hardware counters

```
% export VT_METRICS=PAPI_FP_OPS:PAPI_L2_TCM
```

- Set a new file prefix

```
% export VT_FILE_PREFIX=bt_4_papi
```

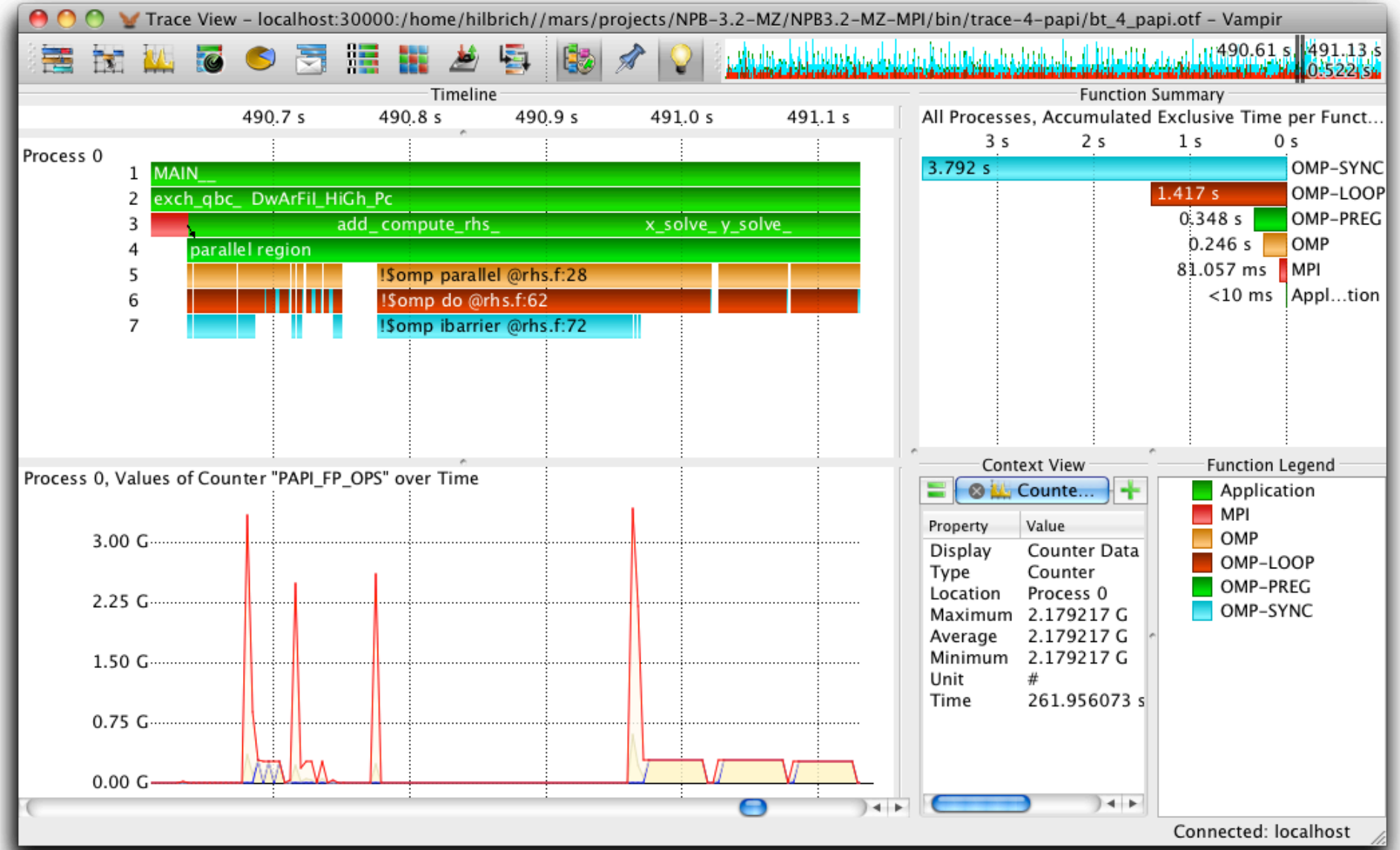
Add both to your jobscript, keep filtering and buffer size environmentals

- Resubmit jobscript

```
bsub < run.lsf
```

```
msub run.msub
```

Hands-on: NPB – Filter & Counter Trace



VampirTrace Run-Time Options

- control options by environment variables:

– VT_PFORM_GDIR	Directory for final trace files
– VT_PFORM_LDIR	Directory for intermediate files
– VT_FILE_PREFIX	Trace file name
– VT_BUFFER_SIZE	Internal trace buffer size
– VT_MAX_FLUSHES	Max number of buffer flushes
– VT_MEMTRACE	Enable memory allocation tracing
– VT_MPICHECK	Enable MPI checking
– VT_IOTRACE	Enable I/O tracing
– VT_MPITRACE	Enable MPI tracing
– VT_FILTER_SPEC	Name of filter definition file
– VT_GROUPS_SPEC	Name of grouping definition file
– VT_METRICS	PAPI counter selection

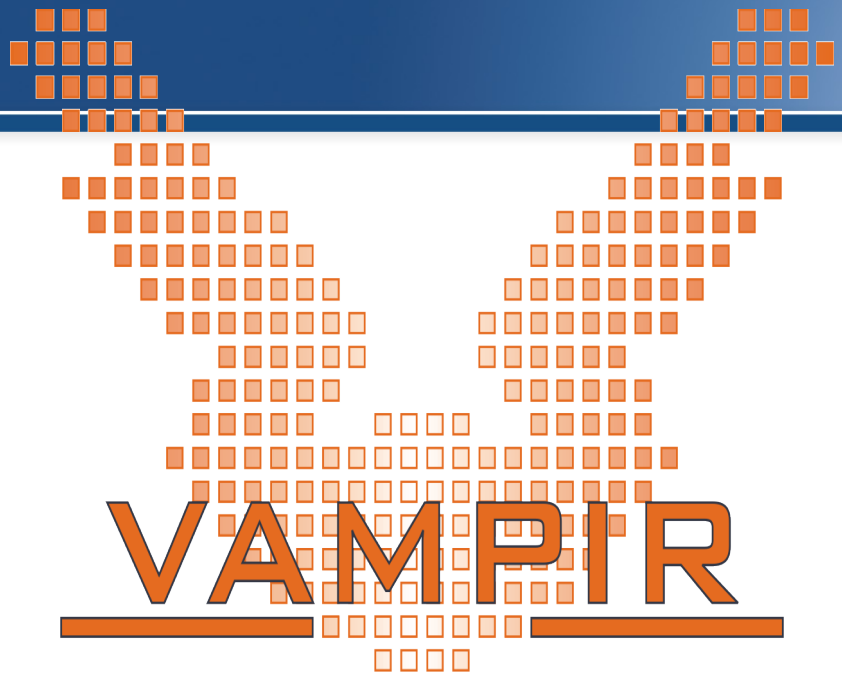
Conclusions and Outlook

- performance analysis very important in HPC

- use performance analysis tools for profiling and tracing
- do not spend effort in DIY solutions,
  e.g. like printf-debugging

- use tracing tools with some precautions
  - overhead
  - data volume

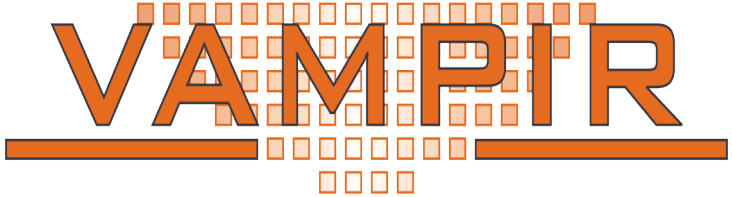
- let us know about problems and about feature wishes
- vampirsupport@zih.tu-dresden.de



Vampir and VampirTraces are available at http://www.vampir.eu and http://www.tu-dresden.de/zih/vampirtrace/ ,

get support via vampirsupport@zih.tu-dresden.de

Acknowledgement:



Staff at ZIH - TU Dresden:

Ronny Brendel, Holger Brunst, Jens Doleschal, Ronald Geisler, Daniel Hackenberg, Michael Heyde, Tobias Hilbrich, Rene Jäkel, Matthias Jurenz, Michael Kluge, Andreas Knüpfer, Matthias Lieber, Holger Mickler, Hartmut Mix, Matthias Müller, Wolfgang E. Nagel, Reinhard Neumann, Michael Peter, Heide Rohling, Johannes Spazier, Michael Wagner, Matthias Weber, Bert Wesarg