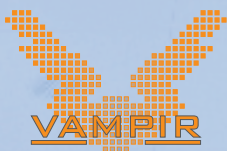
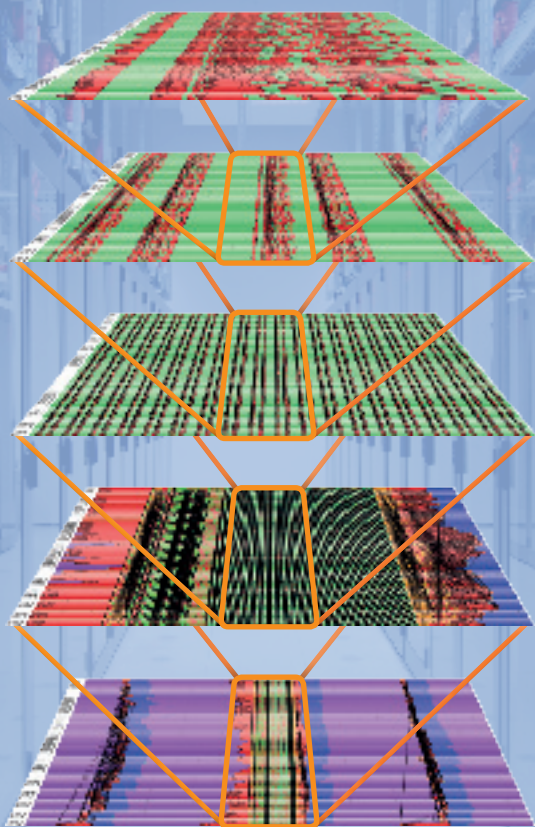


Performance Analysis

www.vampir.eu

www.score-p.org

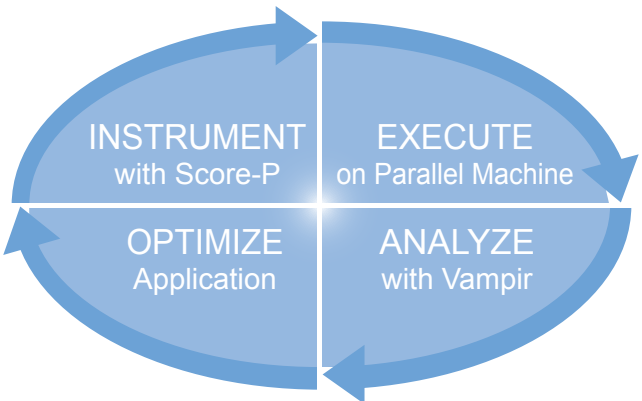


Goals

- Visualize application behavior
- Examine application execution during a given time in a given process or thread
- Investigate communication patterns of the application executed on a real system
- Identify
 - Bottlenecks
 - Load imbalances
 - Single core inefficiencies
 - I/O bottlenecks
- Understand performance impacts
- Optimize parallel applications

Workflow

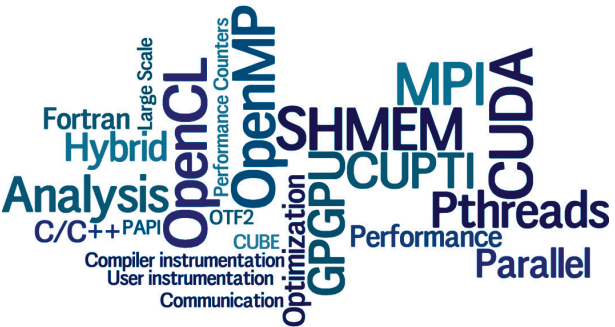
- Prepare application (with symbols), insert extra code (probes/hooks)
- Collection of relevant data as input for the performance analysis
- Calculation of metrics, identification of performance metrics
- Presentation of results in an intuitive/understandable form
- Modifications intended to eliminate/reduce performance problems



Score-P

- Monitor and record performance data of HPC applications
- Tool-suite for profiling, event tracing, and online analysis

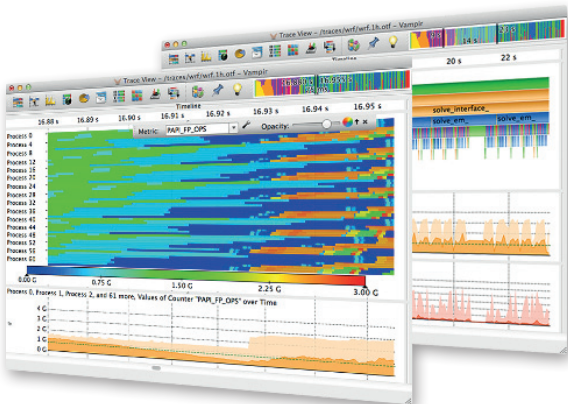
Download: <http://www.score-p.org>



Vampir

- Scalable trace visualization
- In-depth analysis of parallel application runtime behavior, inter-process communication, synchronization, and I/O
- Statistics dynamically adapt to the selected time range

Download: <https://www.vampir.eu/downloads/demo>



Instrumentation Guide

Score-P

- Install Score-P

```
tar xvzf scorep-1.4.2.tar.gz
cd scorep-1.4.2
./configure -prefix=<inst_path>
make
make install
```

- Instrument your application

```
export PATH=\
    <inst_path>/bin:$PATH
cd test/jacobi/hybrid/C
scorep mpicc -fopenmp\
    jacobi.c main.c -o\
    jacobi.scorep
```

- Create a profile with full instrumentation

```
export SCOREP_EXPERIMENT_DIRECTORY=\
    scorep-jacobi
export OMP_NUM_THREADS=4
mpirun -np 4 ./jacobi.scorep
```

- Create initial filter file

```
echo "SCOREP_REGION_NAMES_BEGIN \
    EXCLUDE" >scorep.filt
export SCOREP_FILTERING_FILE=scorep.filt
```

- Incrementally refine the filter file using the output of the scorep-score tool and re-run the application until the overhead is acceptable

```
scorep-score -r -f ./scorep.filt \
    scorep-jacobi/profile.cubex
mpirun -np 4 ./jacobi.scorep
```

- For an in-depth analysis, generate a trace with the filter applied

```
export SCOREP_ENABLE_TRACING=true
export SCOREP_ENABLE_PROFILING=false
mpirun -np 4 ./jacobi.scorep
```

Analysis Guide

Vampir

- Install Vampir

```
./vampir-8.4.1-demo-linux-x86_64-\  
setup.bin --instdir=<inst_path>
```
- Following the Score-P example, you have a trace in the directory scorep-jacobi
- Open your trace file in Vampir for performance analysis

```
vampir scorep-jacobi/traces.otf2
```

The screenshot displays the Vampir Trace View interface. At the top, a window title bar reads "Trace View - /Volumes/Data/Traces/homepage/Large/wrf.otf - Vampir". Below the title bar is a horizontal bar chart showing "Accumulated Exclusive Time per Function Group" from 0s to 208.364 s. The main area features a pie chart titled "Function Summary" with three segments: DYN [1,910.33 s (57.3%)], PHYS [997.876 s (29.93%)], and WRF [398.129 s (11.94%)]. Callout boxes provide context: "Overview of trace data" points to the pie chart, "Profile information for guidance" points to the DYN segment, and "Use context menus for further options" points to the pie chart area. Below the pie chart is a "Timeline" view showing "Process 0" through "Process 15" over a 200-second period. A callout box "Zoom for detailed investigation" points to the timeline. The bottom of the interface shows a status bar.

Contact

Score-P CONTACT

E-mail: support@score-p.org

Web: www.score-p.org

Vampir CONTACT/EU

GWT-TUD GmbH

E-mail: sales@vampir.eu

Web: www.vampir.eu

Vampir CONTACT/U.S.

ParaTools, Inc.

E-mail: info@paratools.com

Web: www.paratools.com
