Message Passing with MPI

PPCES 2018

Joachim Protze
IT Center

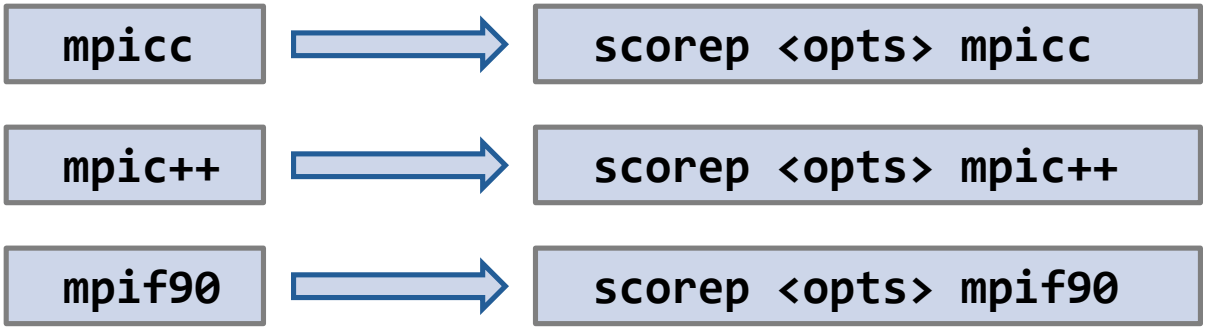
Slides by Hristo Iliev

Performance Tools

- **MPI programs do not always behave as expected**
  - → Communication delays
  - → Synchronisation overhead
  - → Inefficient communication patterns
- **Tools exist for profiling and tracing the program execution**
  - → Profiling
    - → How often certain functions are called?
    - → How much time is spent in certain parts?
    - → How much data is exchanged in certain parts?
  - → Tracing

Code Instrumentation with Score-P

- **Code first has to be instrumented accordingly:**
  - → Recompile with instrumentation



  - → When run, the instrumented binary produces trace files in OTF format.
- **Instrumentation type**
  - → `--compiler` compiler assisted instrumentation (default)
    very detailed; huge trace files
  - → `--user --nocompiler` manual tracing using Score-P API
  - → `--mpp=mpi` traces MPI events (auto)
  - → `--thread=openmp` traces OpenMP parallel constructs (auto)

Score-P: Code Instrumentation

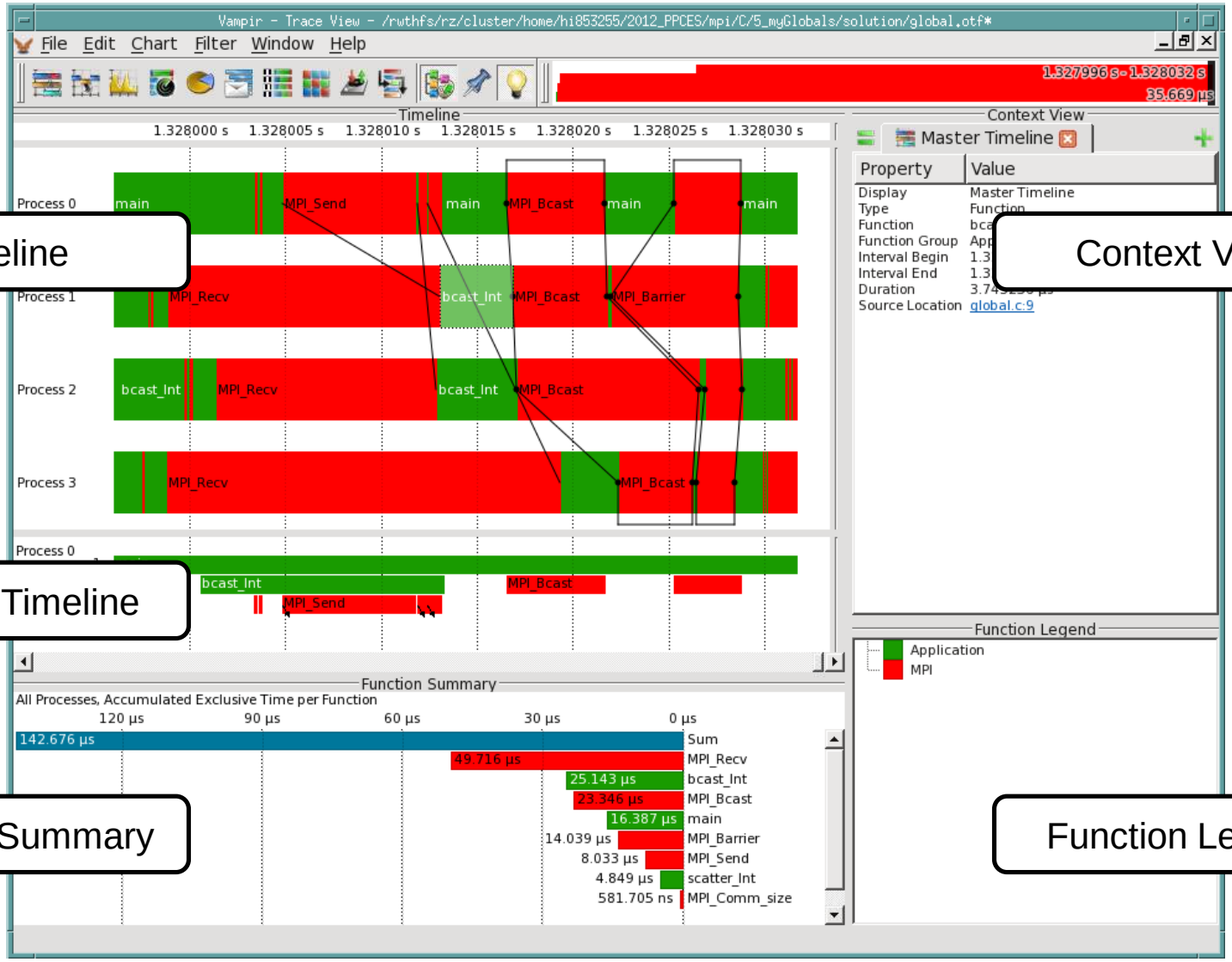
- **Score-P is controlled by many environment variables**
  - → **`SCOREP_TOTAL_MEMORY`** Total memory in bytes for Score-P per process (default: 16M)
  - → **`SCOREP_FILTERING_FILE`** Name of the filter rules file (if any)
  - → **`SCOREP_METRIC_PAPI`** PAPI metrics to record
  - → **`SCOREP_ENABLE_PROFILING`** enable profiling (default: on)
  - → **`SCOREP_ENABLE_TRACING`** enable tracing (default: off)
  - → **`SCOREP_EXPERIMENT_DIRECTORY`** directory where the profile and the trace files are to be stored
- **scorep-score can be used to estimate the max trace buffer capacity and the (uncompressed) trace file size given an application profile**

Sample Workflow with Score-P

```
> module load UNITE scorep vampir
> scorep $MPICC -o prog.exe program.c
--- an instrumented executable produced ---
> $MPIEXEC -n 4 -x SCOREP_EXPERIMENT_DIRECTORY=profiling \
           -x LD_LIBRARY_PATH prog.exe
--- program output ---
--- program profile written in profiling/profile.cubex ---
> scorep-score profiling/profile.cubex
Estimated aggregate size of event trace:                   26kB
Estimated requirements for largest trace buffer (max_buf): 11kB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       4097kB
                                                           ^^^^^^

> $MPIEXEC -n 4 -x SCOREP_EXPERIMENT_DIRECTORY=tracing \
           -x SCOREP_ENABLE_PROFILING=0 -x SCOREP_ENABLE_TRACING=1 \
           -x SCOREP_TOTAL_MEMORY=5000kB -x LD_LIBRARY_PATH prog.exe
--- traces written as tracing/traces.otf2 ---
> vampir tracing/traces.otf2
```

Vampir GUI: Trace View



Timeline

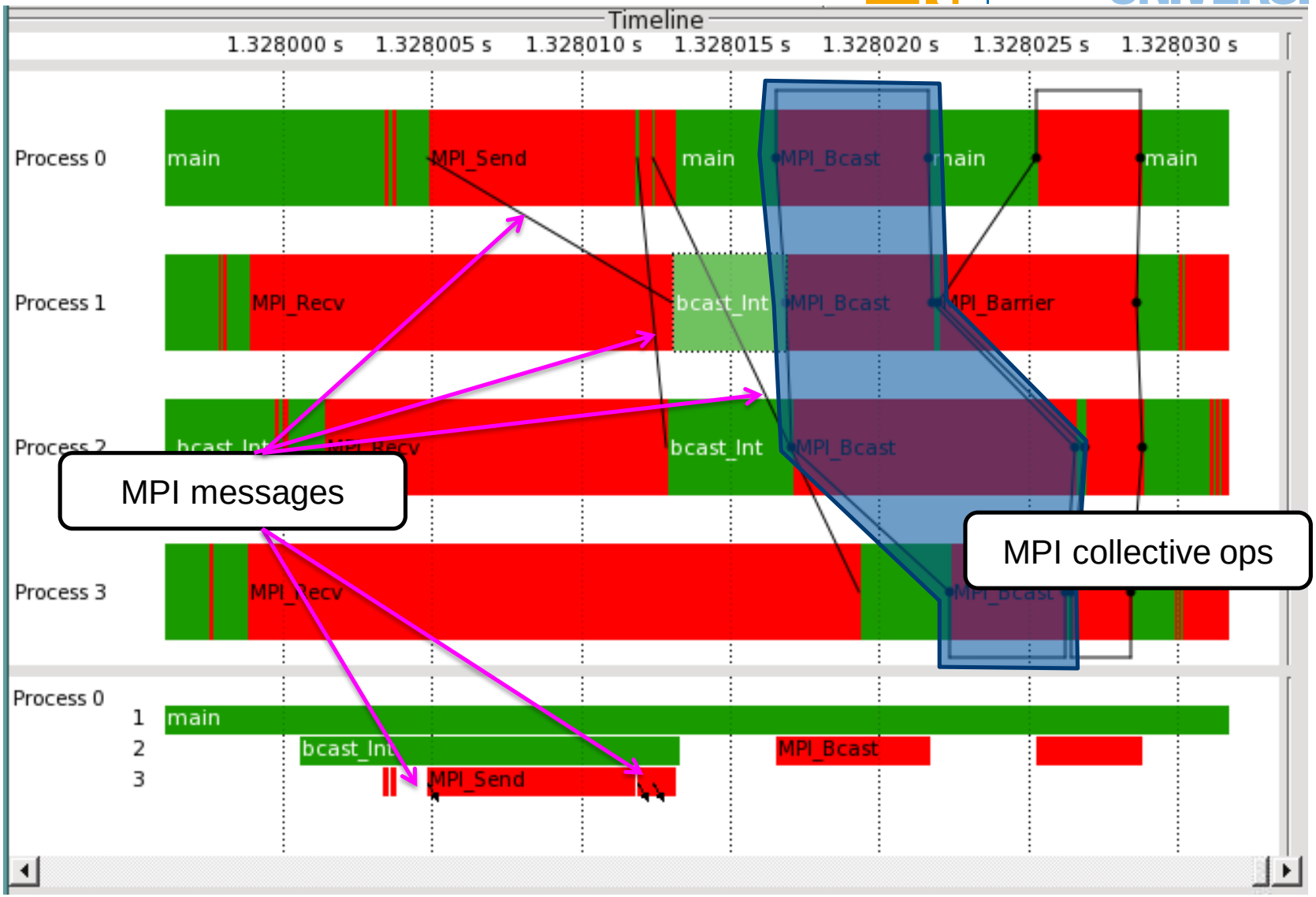
Context View

Process Timeline

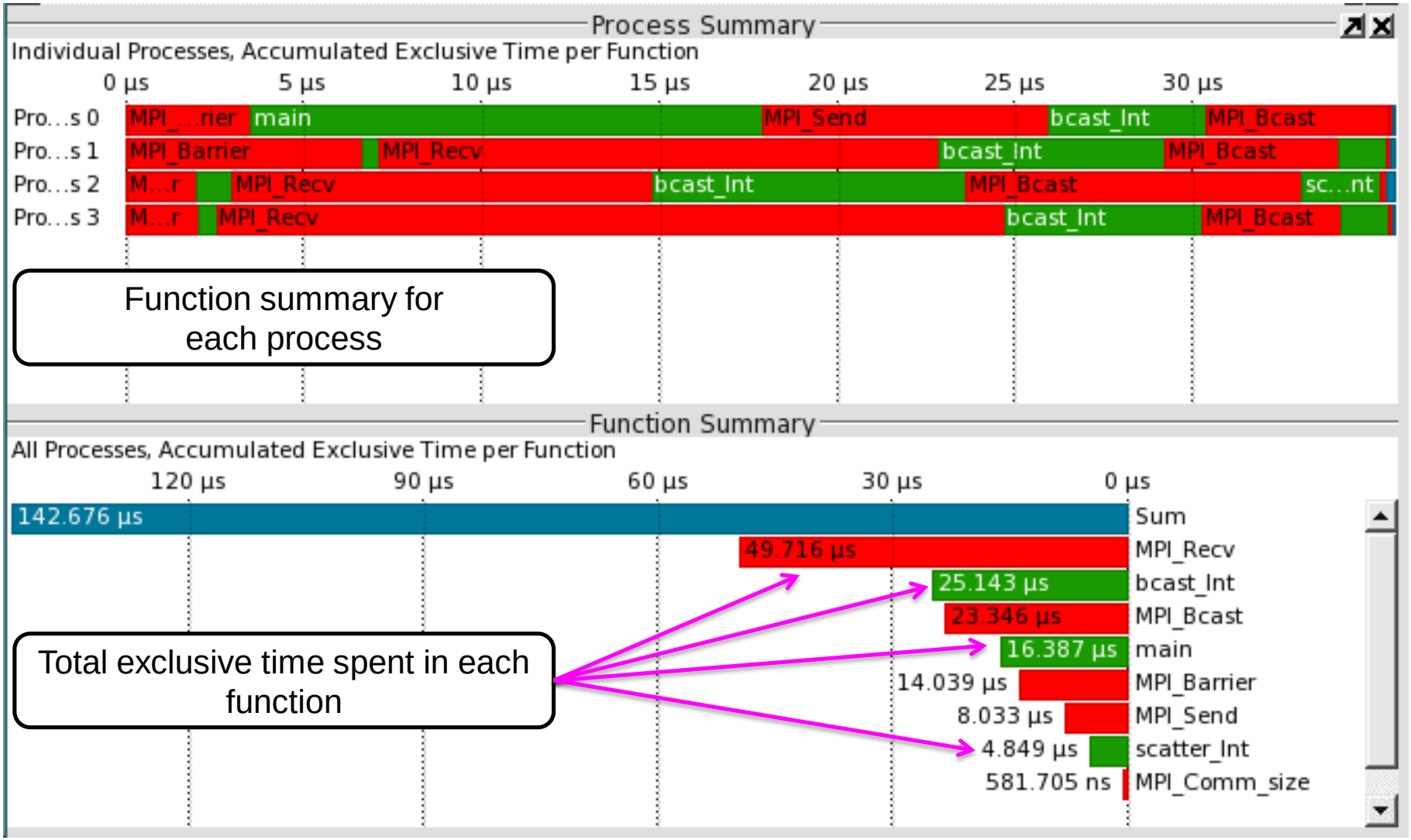
Function Summary

Function Legend

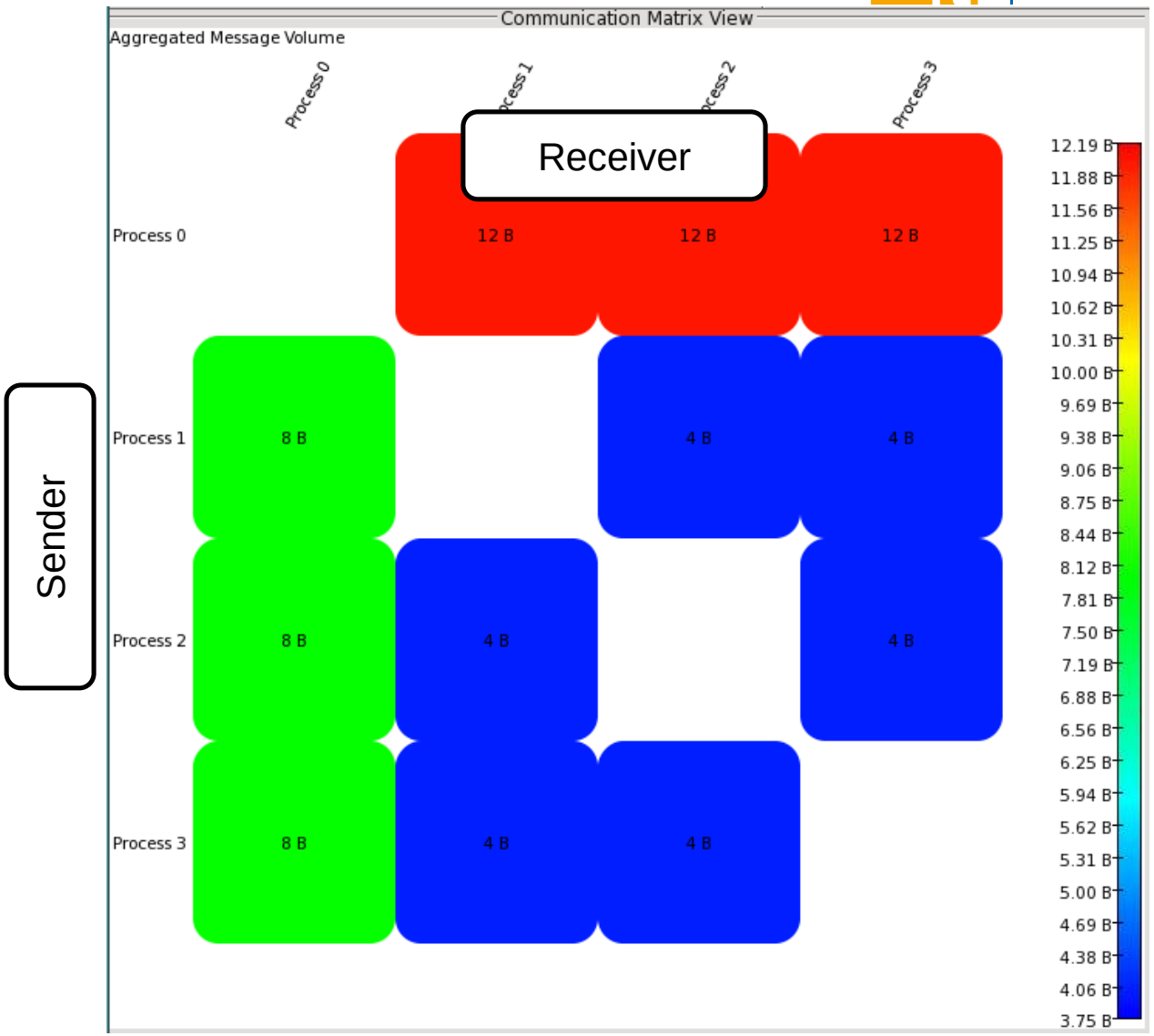
Vampir GUI: Timelines



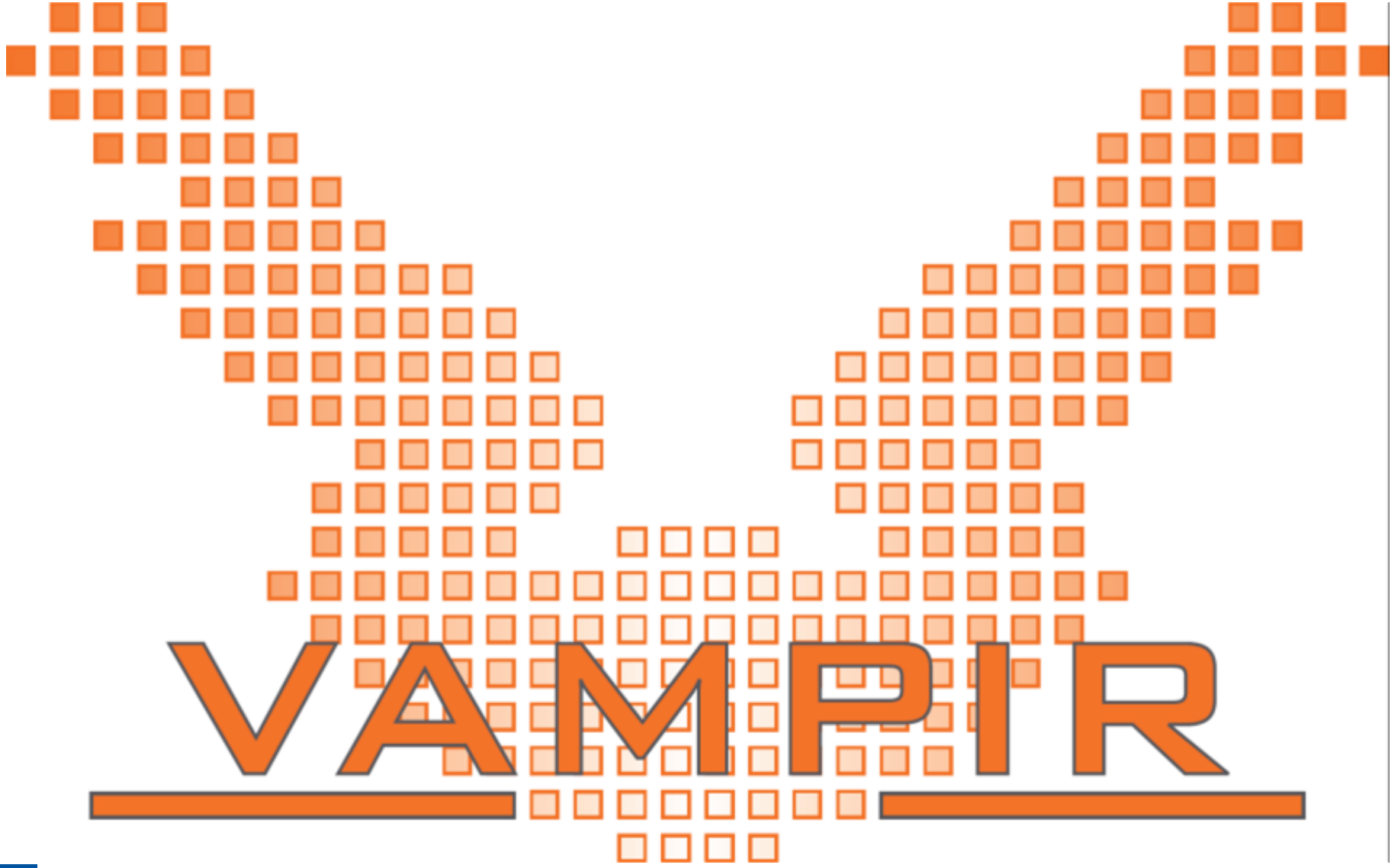
Vampir GUI: Summaries



Vampir GUI: Communication Matrix



Live Demonstration



Thank you for your attention!