Message Passing with MPI

PPCES 2016

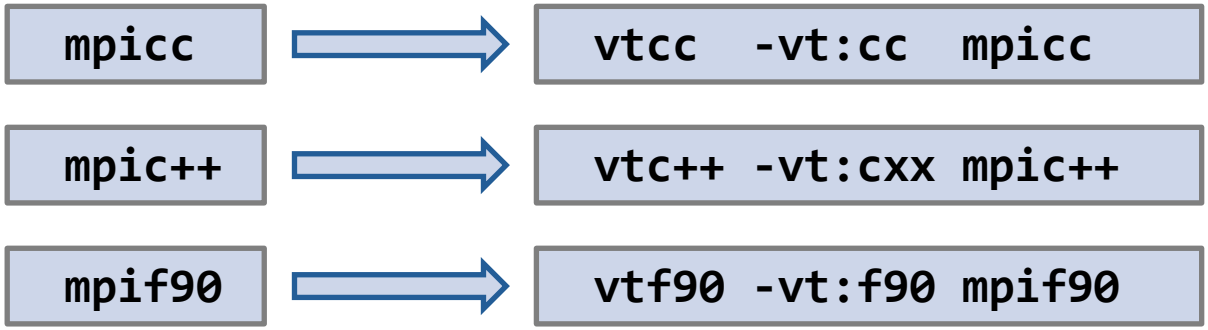
Hristo Iliev
IT Center / JARA-HPC

Vampir Suite

- **Vampir tool suite:**
  - → **Vampir**
    - →visualisation and analysis GUI tool
    - →reads OTF files produced by various tracing tools
    - →commercial
  - → **VampirTrace**
    - →open-source instrumentation tool
    - →tracing library
    - →writes OTF trace files
    - →deprecated in favour of Score-P
- **RWTH Compute cluster: Vampir modules are in the UNITE category**

VampirTrace: Code Instrumentation

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



  - → When run, the instrumented binary produces trace files in OTF format.

- **Instrumentation type selected via `-vt:inst <type>`**
  - → **`compinst`** compiler assisted instrumentation (default)
    all function calls traced; very detailed; huge trace files
  - → **`manual`** manual tracing using VampirTrace API
    traces only MPI events and user-specified events;
    significantly reduced trace file size

VampirTrace: Code Instrumentation

- **VampirTrace is controlled by many environment variables**
  - → **`VT_BUFFER_SIZE`** internal trace buffer size; flushed to the disk when full (default: 32M)
  - → **`VT_FILE_PREFIX`** OTF file prefix (default: executable name)
  - → **`VT_MAX_FLUSHES`** number of trace buffer flushes before tracing is disabled (0 – no limit)
  - → **`VT_SYNC_FLUSH`** synchronised buffer flushes (default: no)

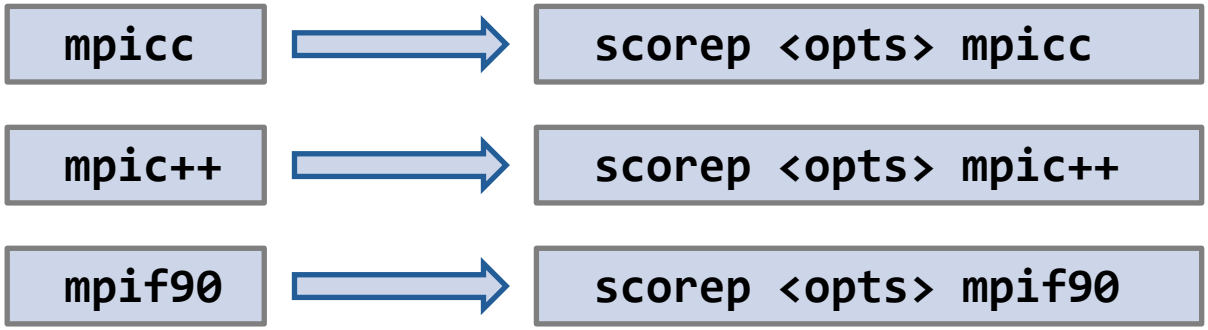
- **Things to be aware of:**
  - → By default buffers are flushed asynchronously and it takes time
  - → Significant skew in program's performance profile possible
  - → No trace written after abnormal program termination

VampirTrace: Sample Session

```
> module load UNITE vampirtrace vampir
> vtcc –vt:cc $MPICC –o prog.exe program.c
--- an instrumented executable produced ---
> mpiexec –n 4 –x VT_MAX_FLUSHES=0 –x LD_LIBRARY_PATH prog.exe
--- program output ---
--- program output ---
--- OTF traces get written in the end ---
> vampir prog.exe.otf
--- Vampir GUI opens ---
```

Score-P: Code Instrumentation

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



  - → Enable tracing: **export SCOREP_ENABLE_TRACING=1**
  - → When run, the instrumented binary produces trace files in OTF format

- **Instrumentation type selection (in <opts>)**
  - → **`--compiler`** compiler assisted instrumentation (default) all function calls traced; huge trace files
  - → **`--nocompiler --user`** traces only MPI events and user events; significantly reduced trace file size

Score-P: Code Instrumentation

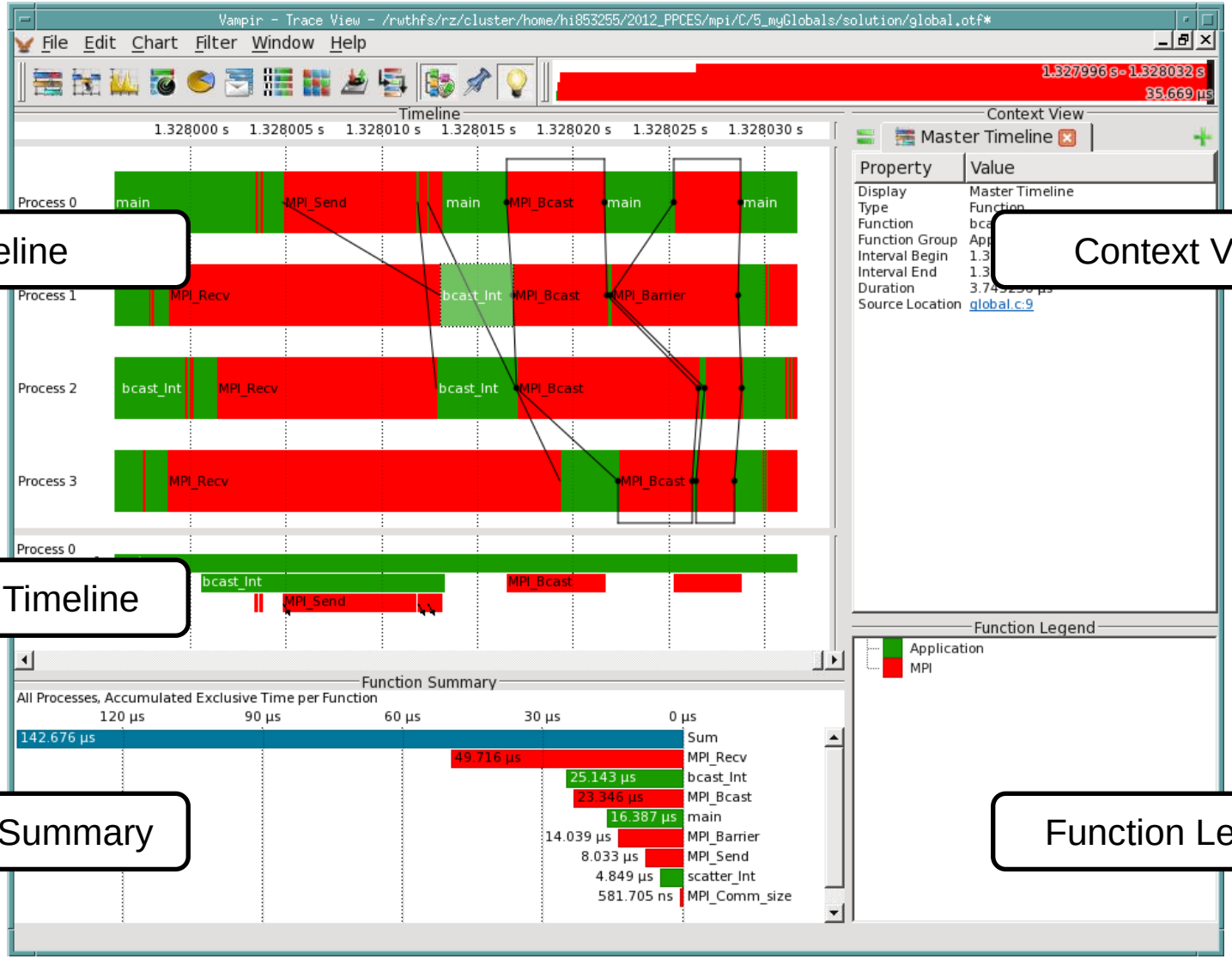
- **Score-P is controlled by many environment variables**
  - → **`SCOREP_TOTAL_MEMORY`** Total memory in bytes for Score-P per process (default: 16M)
  - → **`SCOREP_FILTERING_FILE`** Name of the filter rules file (if any)
  - → **`SCOREP_METRIC_PAPI`** PAPI metrics to record
  - → **`SCOREP_ENABLE_PROFILING`** 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**

Score-P: Sample Session

```
> 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 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

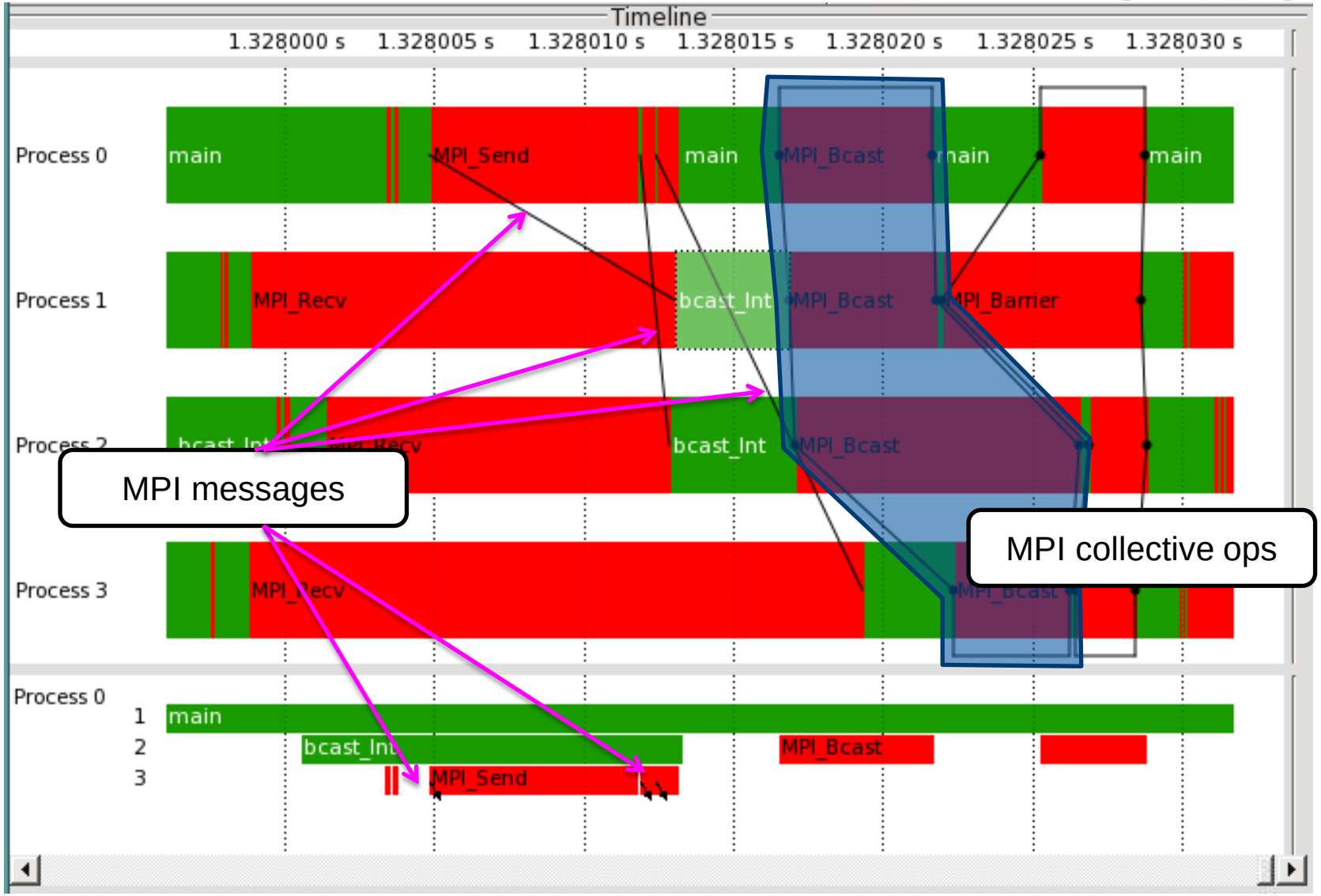
Process Timeline

Function Summary

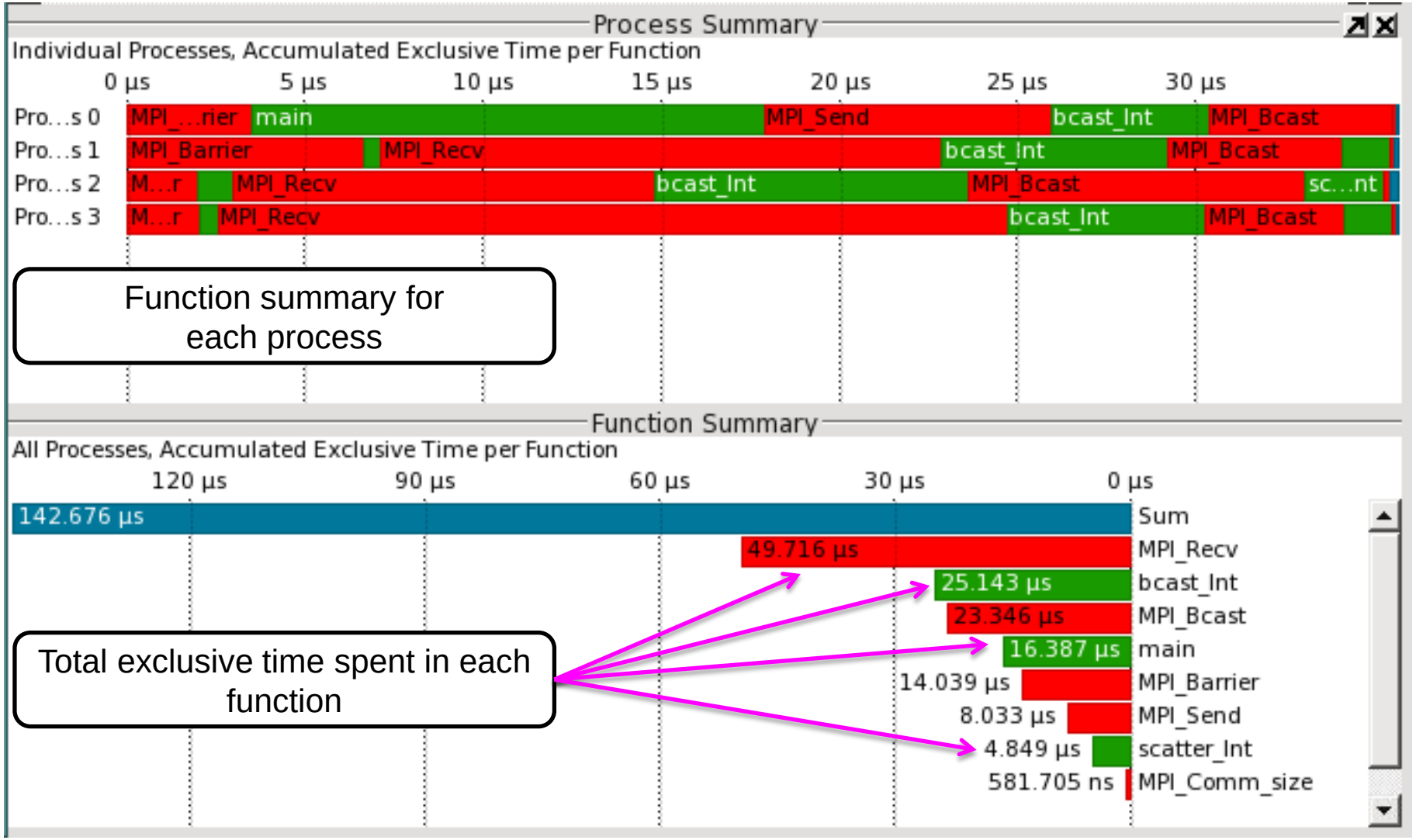
Context View

Function Legend

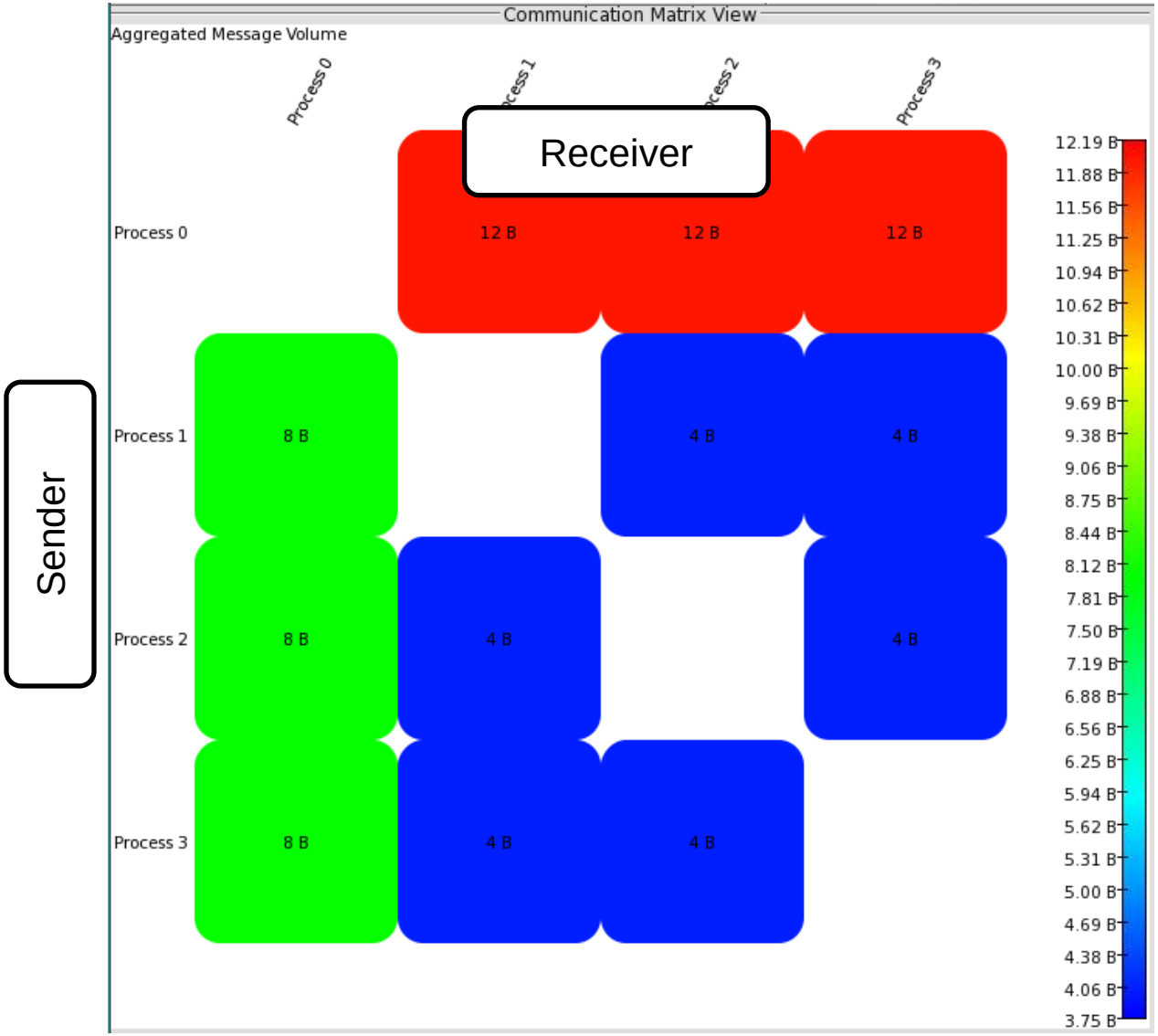
Vampir GUI: Timelines



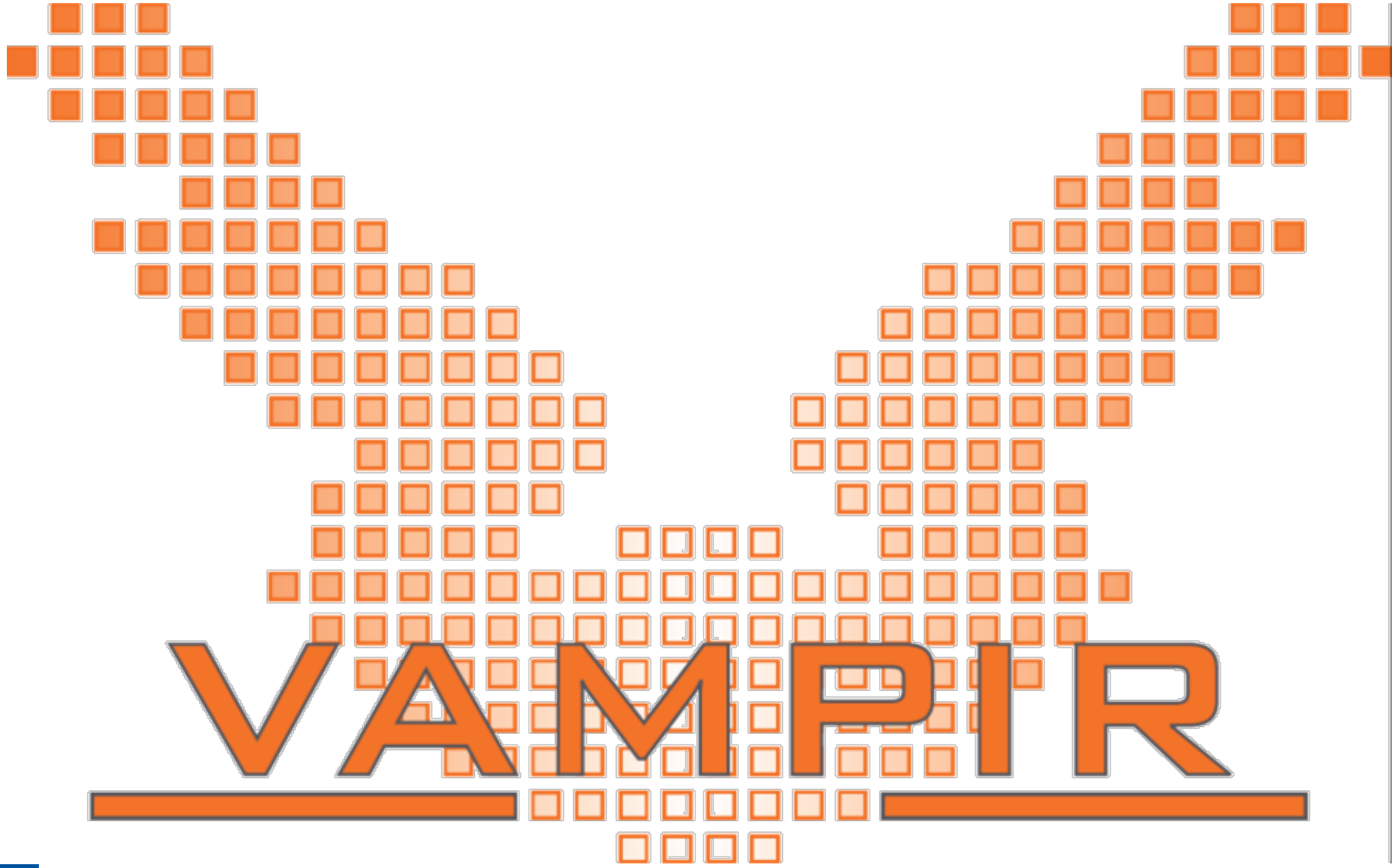
Vampir GUI: Summaries



Vampir GUI: Communication Matrix



Live Demonstration



Thank you for your attention!