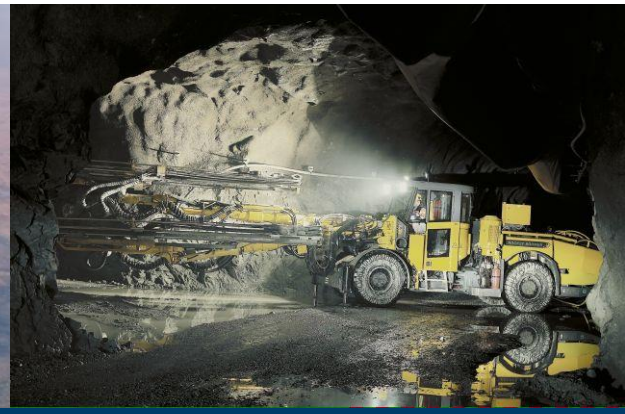


Tracealyzer for Linux

Overview and Getting started

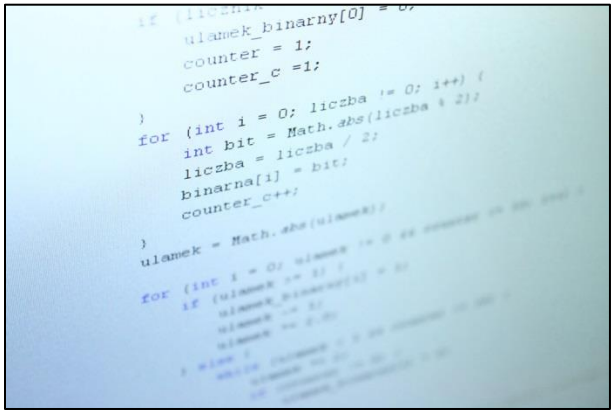


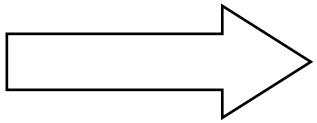


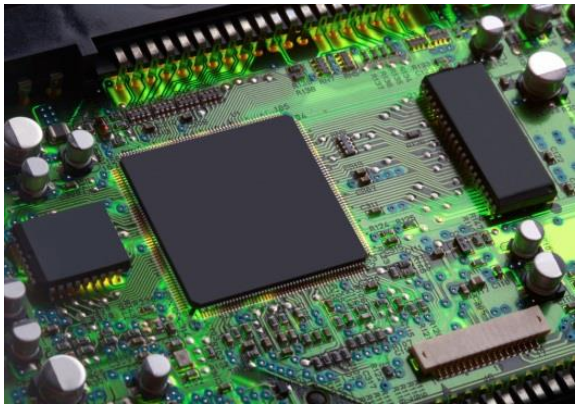
percepio
SENSING SOFTWARE

Software behavior depends on ***timing***

Source code alone does not show the full picture…



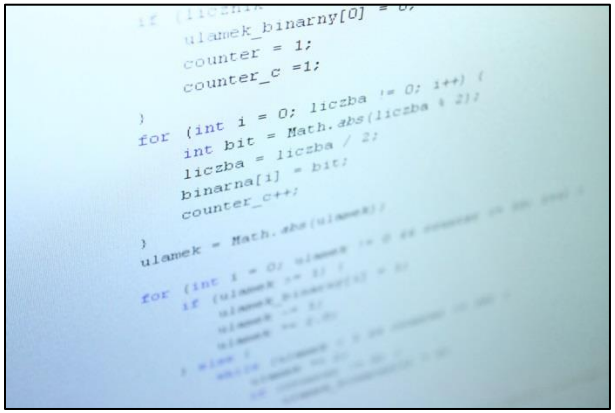


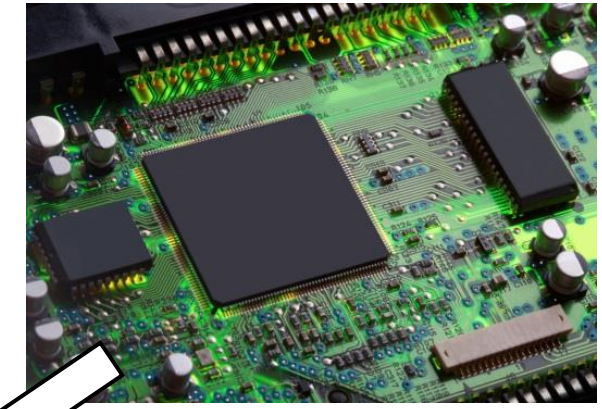


?

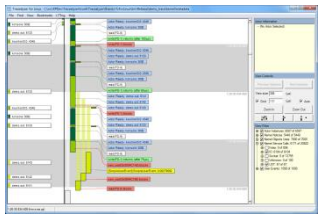
Multi-threaded systems have *context-switches*, i.e., control-flow not visible in source code.
Impacts system performance and dangerous sporadic errors like timeouts or deadlocks.
Depends on code execution times and timing of inputs.

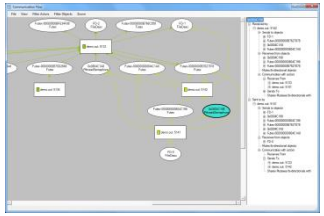
Tracealyzer shows what's going on!

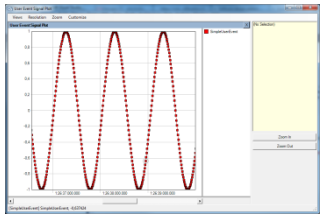




Software-Defined Tracing
+ Easy to use
+ Flexible/configurable
+ Always applicable

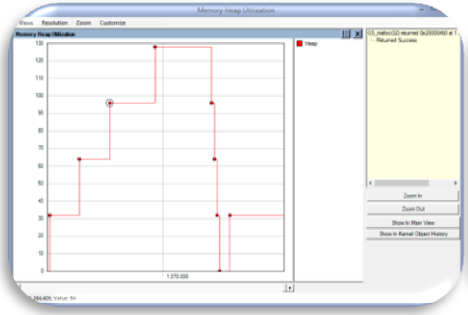
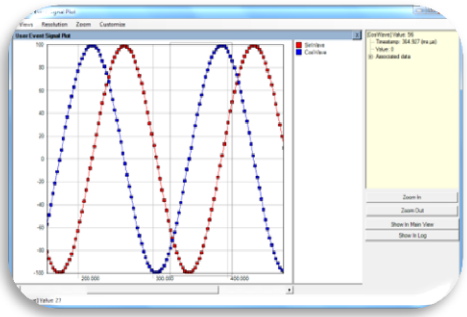
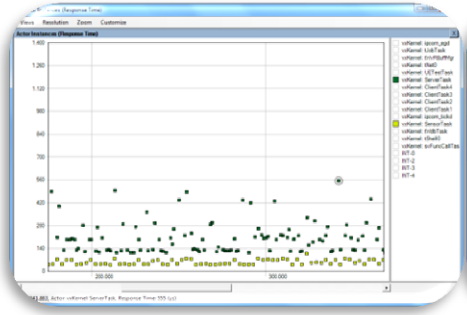
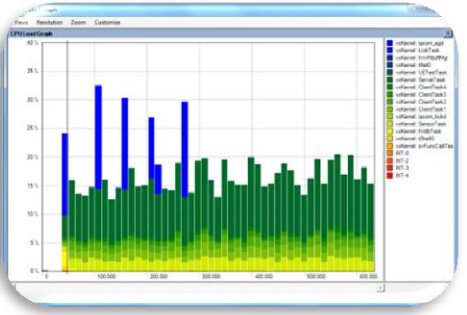
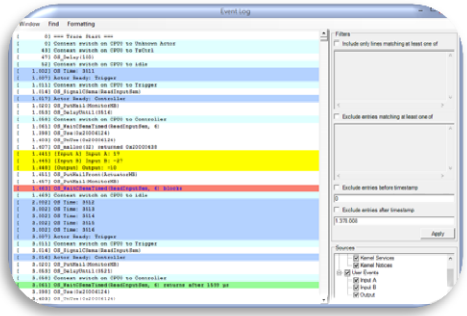
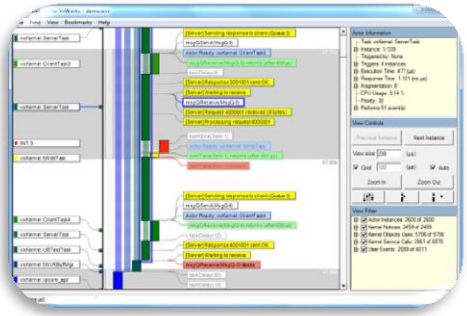
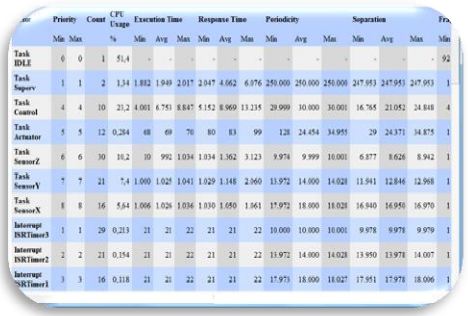


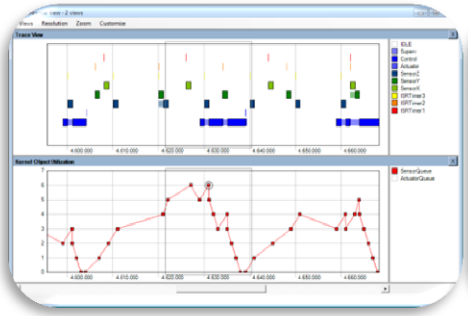
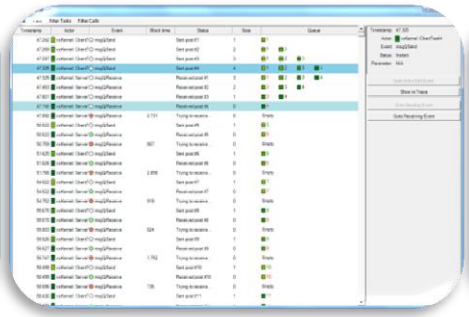
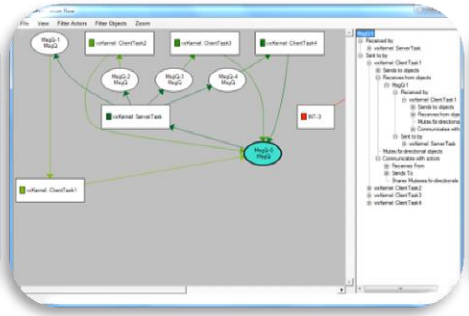
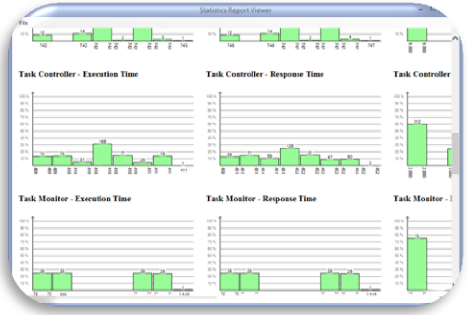




percepio
SENSING SOFTWARE

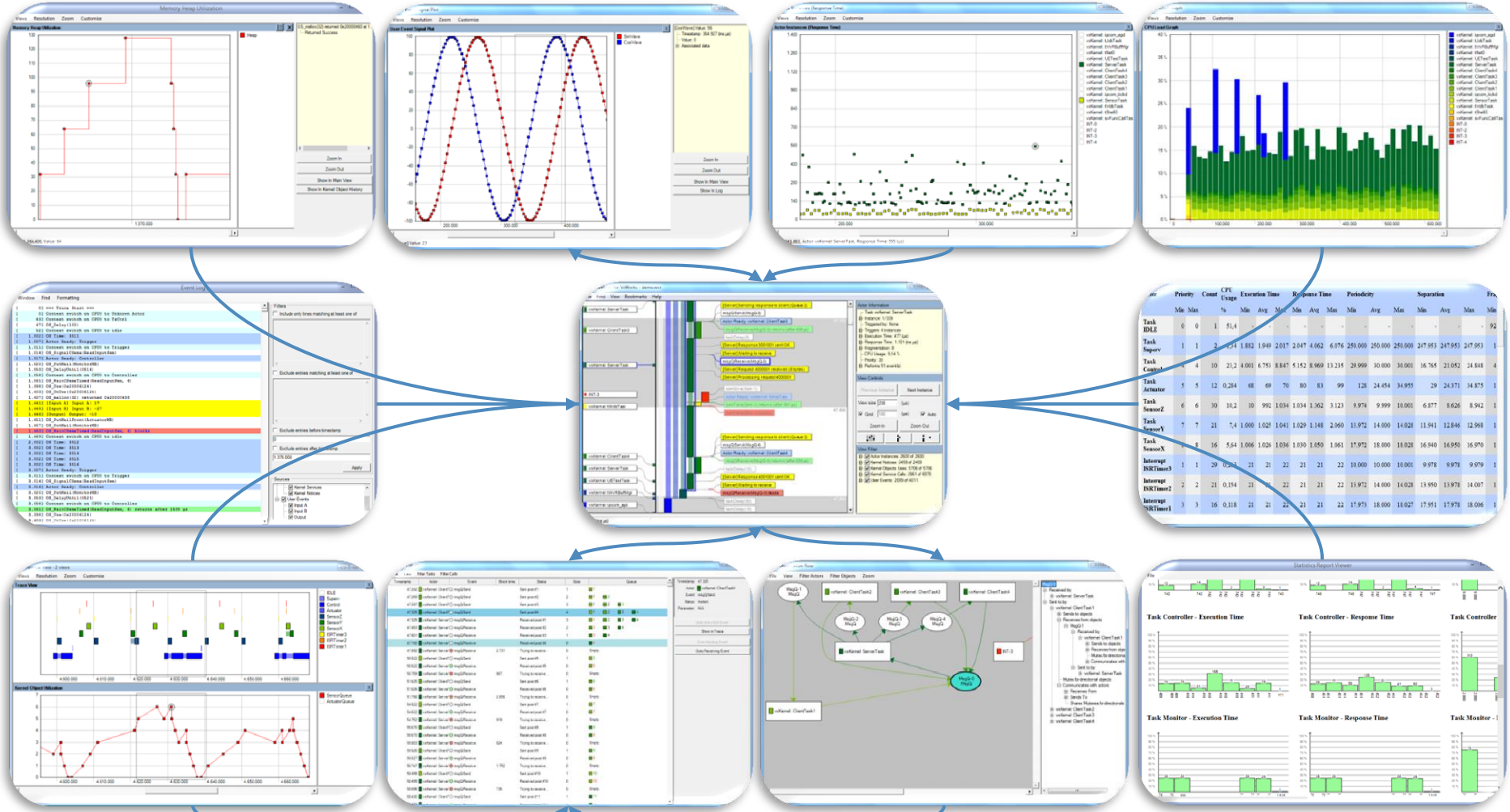
Tracealyzer is about *Visualization*

percepio
SENSING SOFTWARE

Cleverly Interconnected



percepio
SENSING SOFTWARE

Increasing Development Efficiency and Software Performance

Debugging
Detailed real-time behavior

Validation
Code running as intended?

Profiling
RAM and CPU usage?

Training
Learn the software platform

Documentation
Visualize designs or issues

*"In less than 5 days from running the tool, we improved the performance of our graphic rendering engine by **3x!**"*
Terry West, CEO, Serious Integrated Inc.

Serious™

*"Tracealyzer has **doubled our development speed.** Problems that otherwise would take days to solve are obvious with this tool and just a quick fix. We use it all the time."*
Alex Pabouctisids, Lead Firmware Engineer, Flyability.

FLYABILITY

For Leading Companies

*Percepio's tracing tool allowed me to **quickly understand and solve serious multi-threading issues**, that otherwise would have taken least two weeks to analyze. I got started and solved the first issue in a single day. I strongly recommend Percepio's tracing tools."*
Chaabane Malki, Embedded Systems Engineer, CGX Aero

*ABB Robotics is using the first generation Tracealyzer in **all of the IRC5 robot controllers shipped since 2005**. The tool has proven its value many times in all corners of the world."*
Roger Kulläng, Global System Architect, ABB Robotics.

*"In today's tough competition with time-to-market pressure constantly increasing, visualization support is natural for software developers in order to produce software of **higher quality, in shorter time and at a lower cost.** We choose Tracealyzer from Percepio."*
Jörgen Appelgren, R&D Manager, Atlas Copco Rock Drills

*"The many system views of the Tracealyzer from Percepio **makes it easy to quickly find solutions** that we have not seen using (Wind River) System Viewer. The visualization has several advantages over the system viewer and makes it easier to understand system behavior. This tool would be of great use for us."*
Johan Fredriksson, Software Architect, SAAB AB.

For Leading Software Platforms

Tracealyzer for Linux

WIND

Tracealyzer for VxWorks

freeRTOS

FreeRTOS+Trace

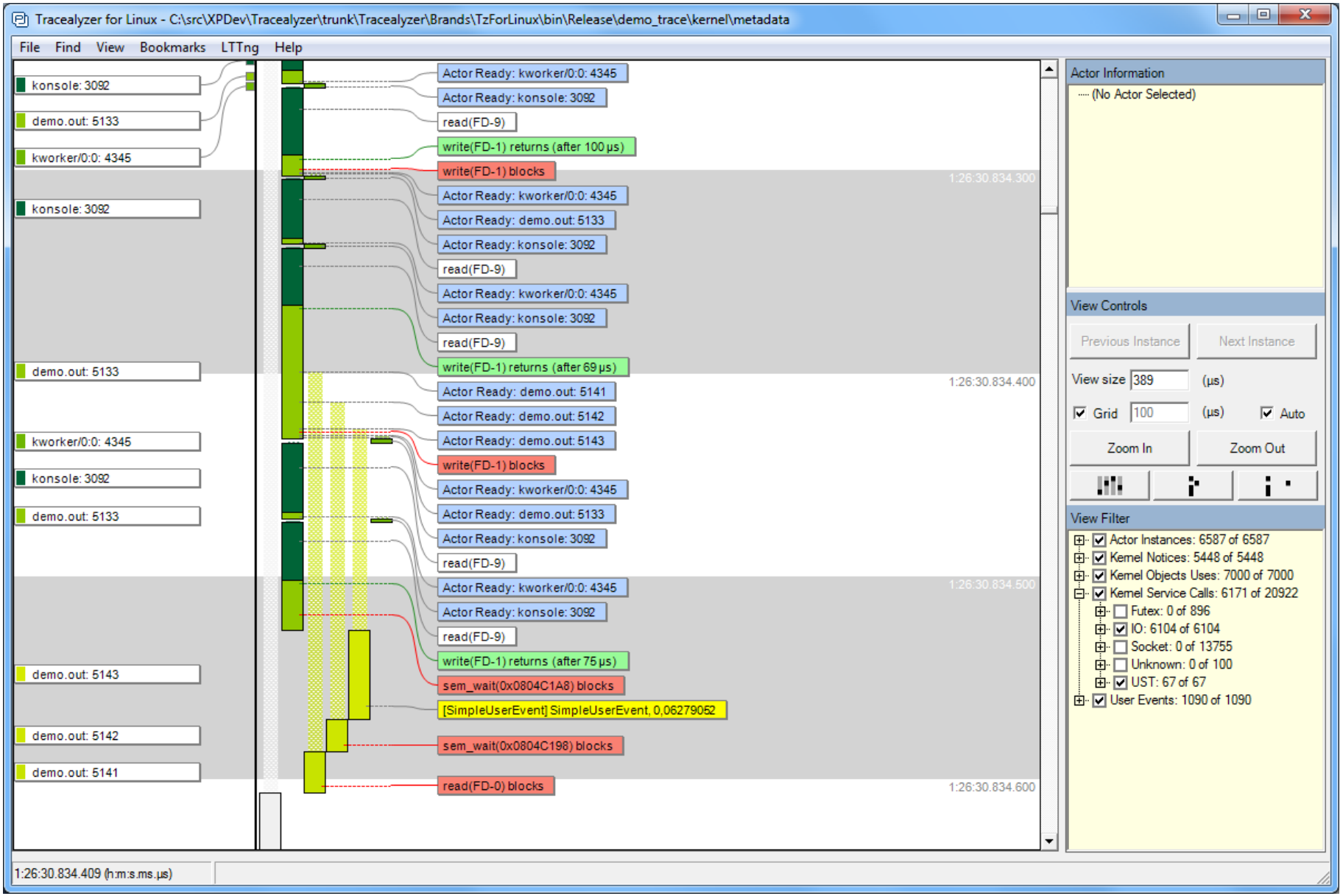
Micriμm

μC/Trace

SafeRTOS+Trace

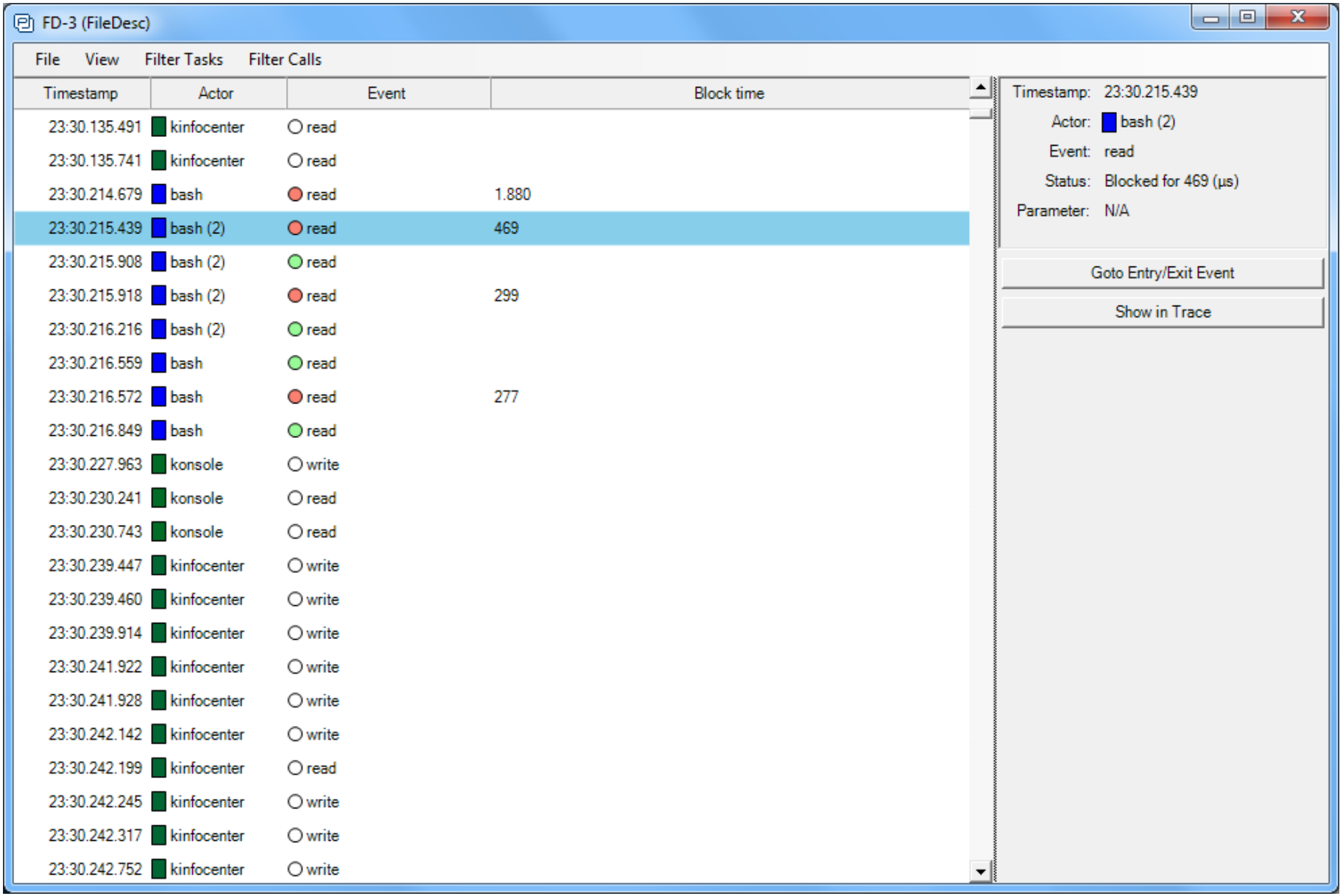
SEGGER

embOS-Trace



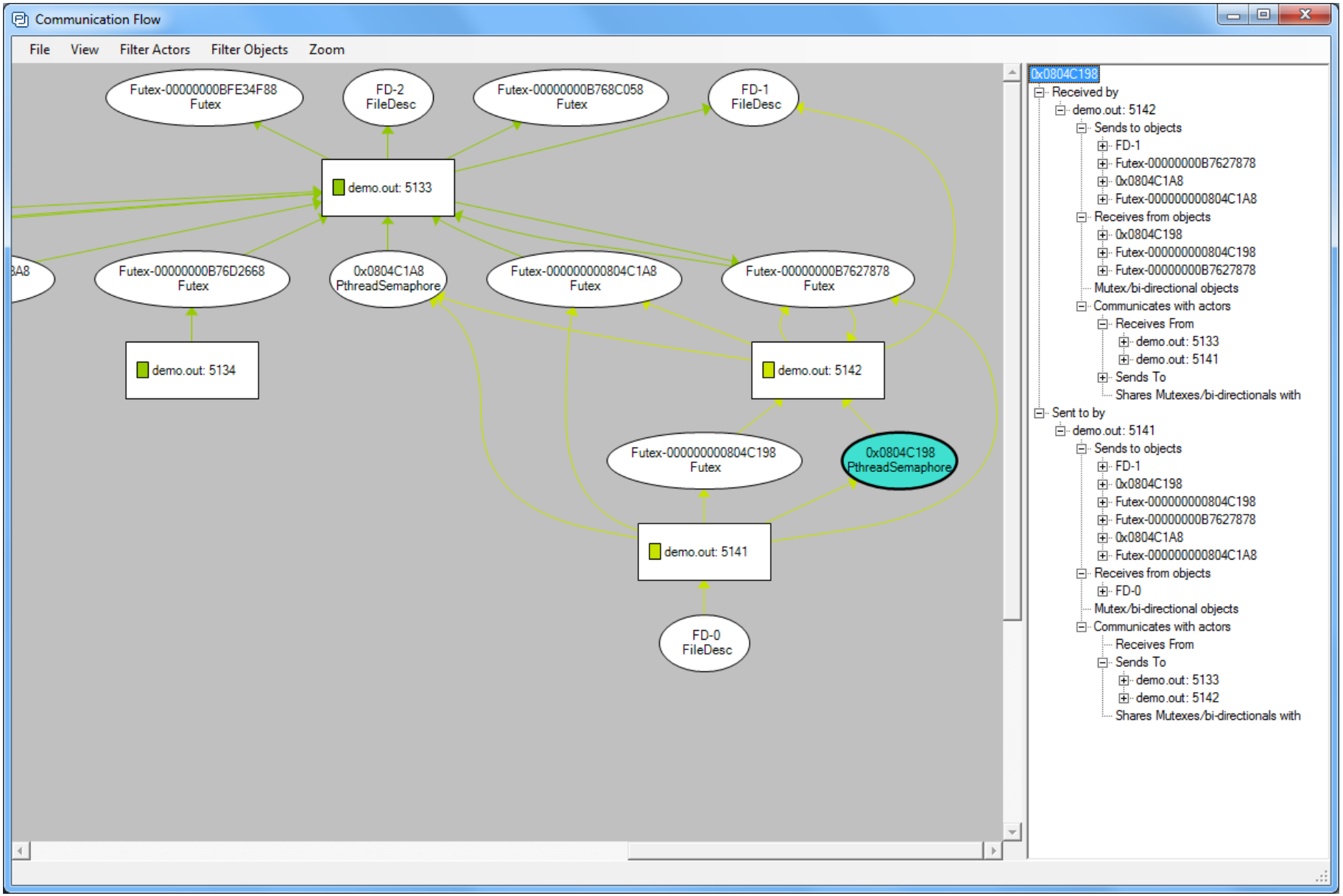
Main Trace View: Vertical time-line showing task scheduling, interrupts, kernel service calls and custom ”User Events”. Filter the display using the ”View Filter” in the bottom right.

percepio SENSING SOFTWARE



Kernel Object History: Double-clicking on a kernel call in the main trace view opens this view, showing the list of events for the same kernel object (e.g., File Descriptor or Semaphore).

percepio SENSING SOFTWARE

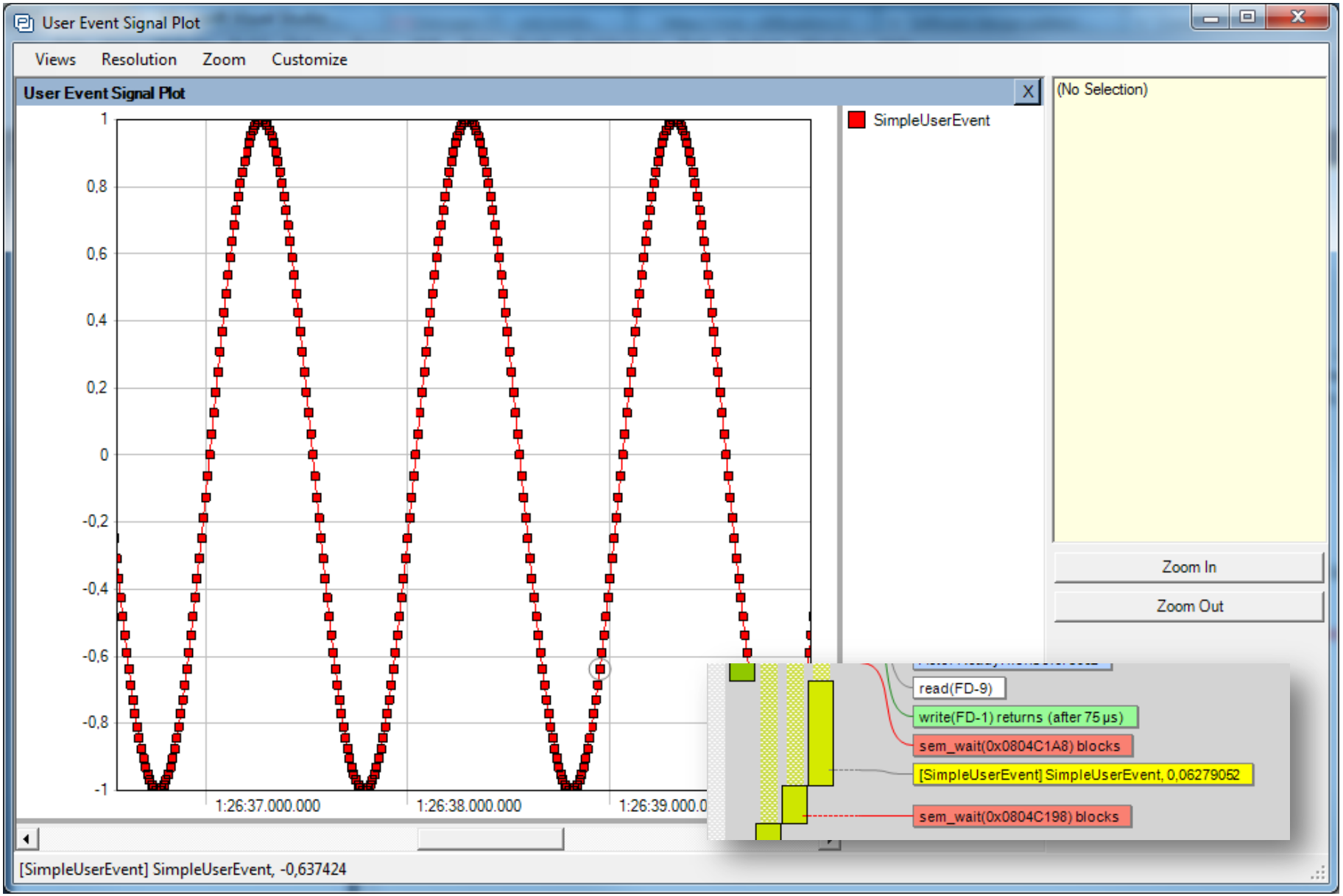


Communication Flow: Auto-generated summary of dependencies between threads and kernel objects. Can be generated for a selected interval, or for the whole trace.

percepio
SENSING SOFTWARE

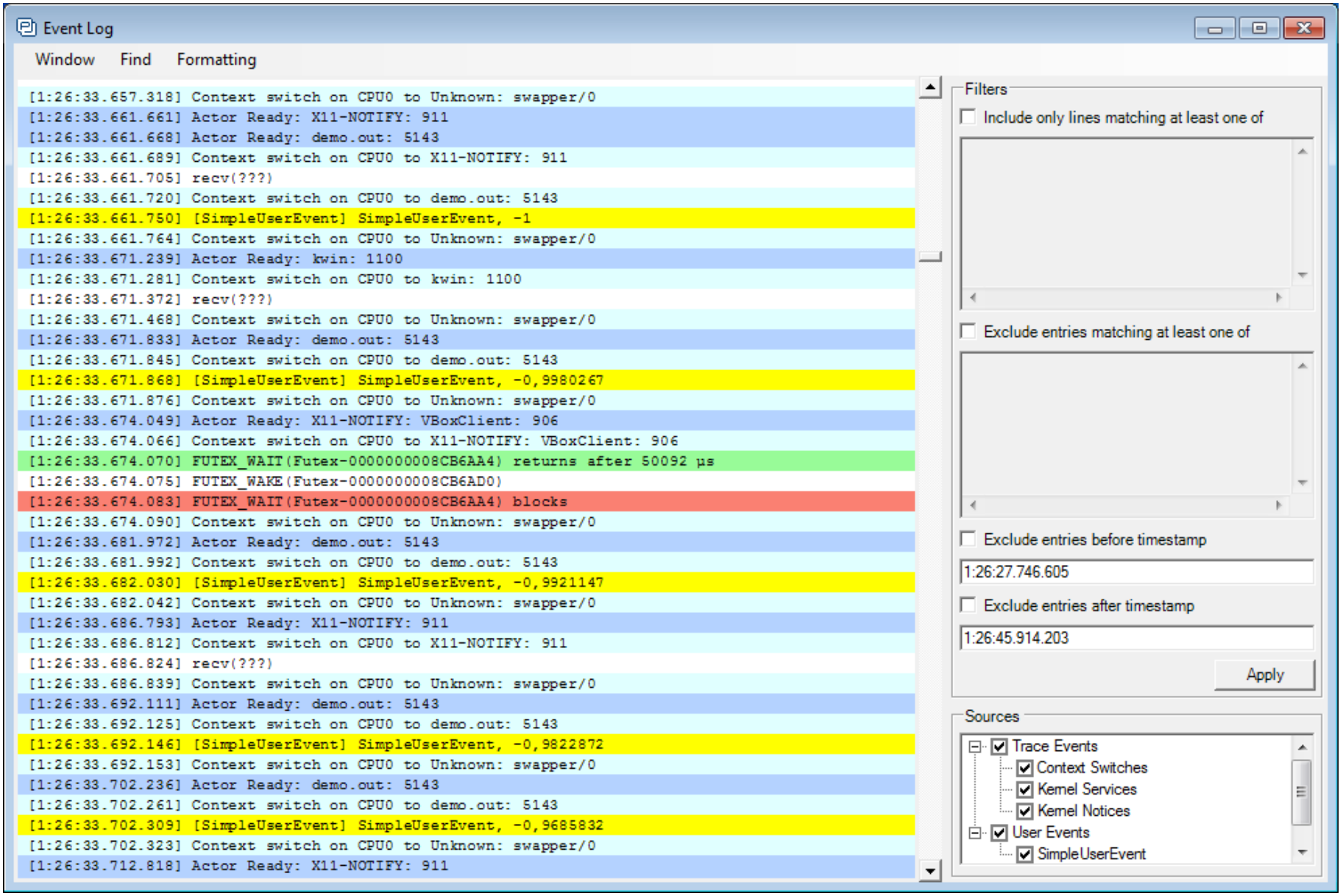
CPU Load Graph: Displays the relative usage of CPU time, per thread and time interval on each processor core. Double-click on intervals to see the details in the main trace view. The interval displayed in the main trace view is indicated using a grey outlined rectangle.

percepio
SENSING SOFTWARE



User Event Signal Plot: Displays a plot of data arguments in ”User Events”, can be used for continuous input signals, state variables, or any data of interest.

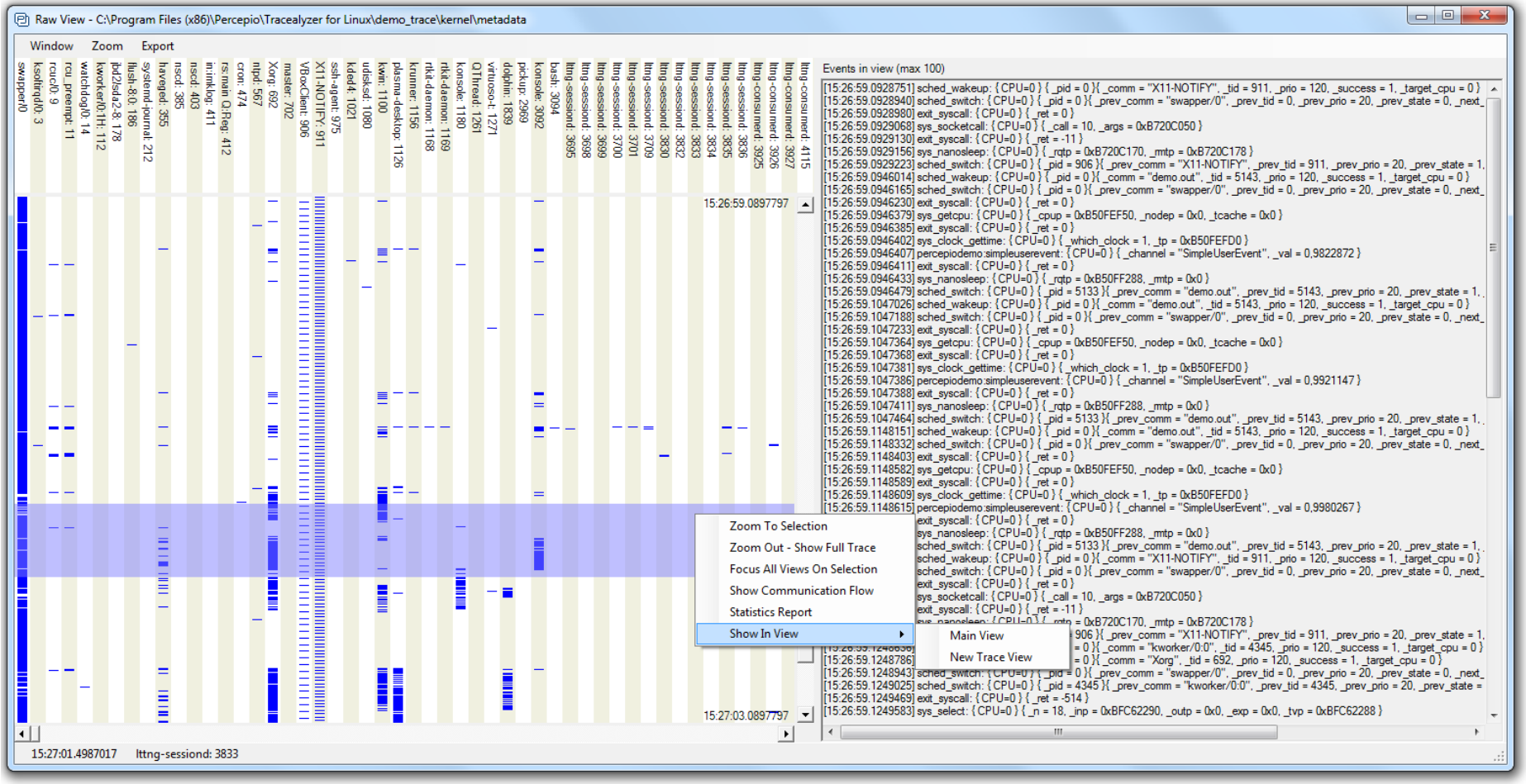
percepio SENSING SOFTWARE



Event Log: Displays a textual list of the event timeline of the main trace view, with filtering and color coding. Exports traces in text format for comparisons and custom analyses.

percepio
SENSING SOFTWARE

Raw Trace View



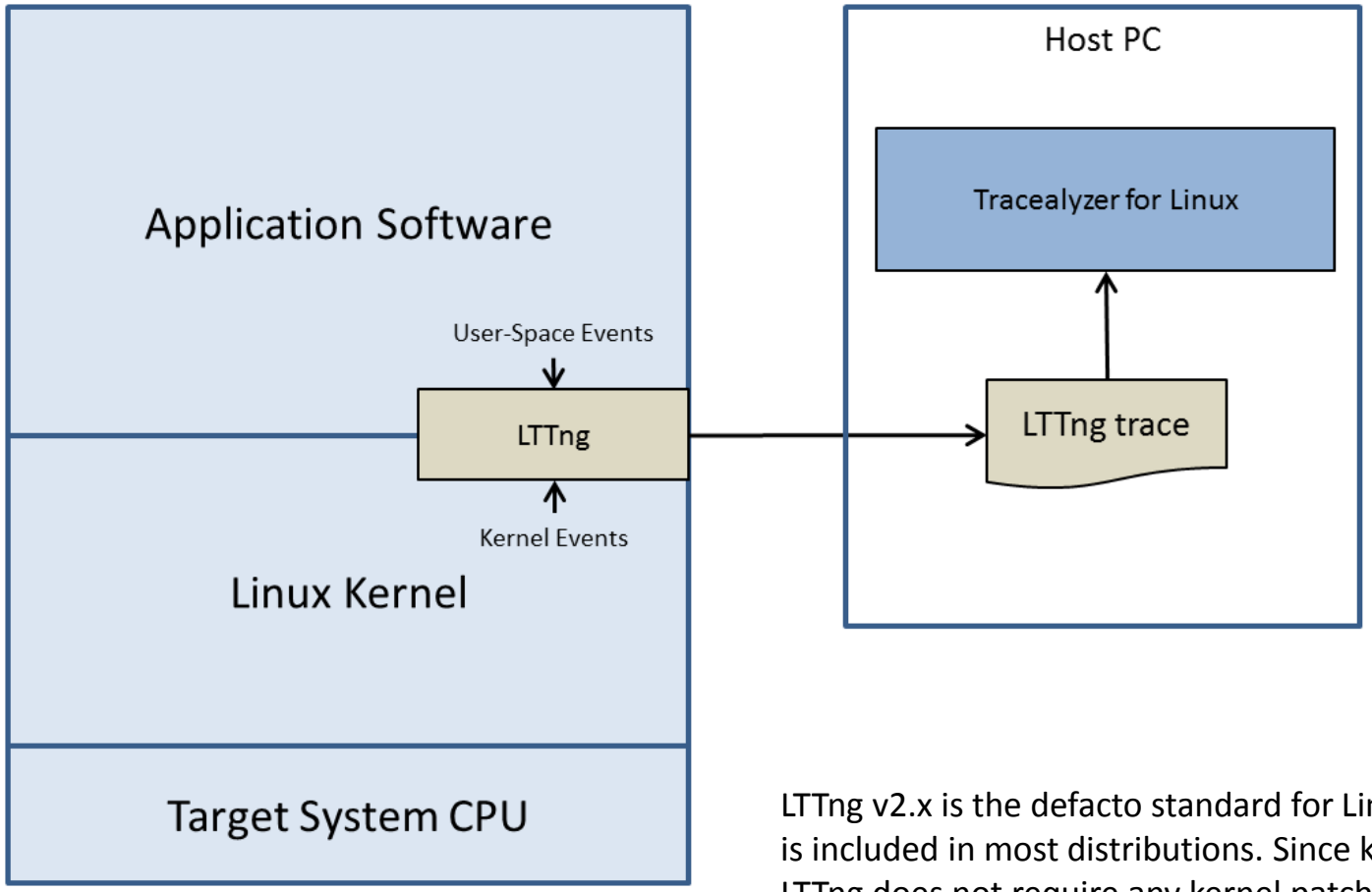
Displays all LTTng events unfiltered, with (almost) no analysis. Works directly from disk to allow for viewing very large traces. You may select and open an interval in the other Tracealyzer views.

percepio
SENSING SOFTWARE

Other Views Provided

- Kernel Object Utilization
  - Watch queue sizes over time
- Kernel Call Intensity
  - Check the rate of kernel calls
- Kernel Blocking Times
  - See blocking on kernel calls
- Scheduling Intensity
  - The rate of context-switches
- Statistics Report
  - Timing and CPU usage
  - Descriptive statistics
  - Timing distributions per job
- Heap Memory Allocation
  - Malloc/free over time
- Actor Instance Graphs
  - Plot the timing of each job

Tracealyzer for Linux relies on LTTng



LTTng v2.x is the defacto standard for Linux tracing and is included in most distributions. Since kernel v.2.6.38 LTTng does not require any kernel patching. But it works also on older kernels from v2.6.32.

Tracing with LTTng

- Install LTTng
  - See https://lttng.org/docs/#doc-installing-lttng
- Create a trace session
  - `lttng create my-session`
- Select what kernel events to enable
  - `lttng list --kernel // shown the available kernel events`
  - `lttng enable-event --kernel sched_switch[, ...] [--all]`
  - "sched_switch" should always be included
- Select any userspace events to enable
  - `lttng list --userspace // shown the available userspace events`
  - `lttng enable-event --userspace [process:tracepoint] [--all]`
- Start tracing
  - `sudo lttng start // default output is ~/lttng-traces`
- Stop tracing
  - `sudo lttng stop`
  - `sudo lttng destroy // closes the trace session`
- Learn more about LTTng at https://lttng.org/

Running on Linux

- Running Tracealyzer on Linux requires **Mono**, the Open Source .NET environment.
  - Check if already installed (mono --version)
  - Otherwise get it from http://www.mono-project.com/
- Download TracealyzerForLinux-2.7.5.tgz or later
  - From http://percepio.com/tz/downloads/
- Extract the .tgz archive to any suitable location
- Open a console and run mono TzForLinux.exe
  - You get 30 days free evaluation on registration
- To open an LTTng trace (a directory of files), select *File* -> *Open* and then locate the ”metadata” file.

Creating ”User Events” from your code

Old detailed method: LTTng tracepoint

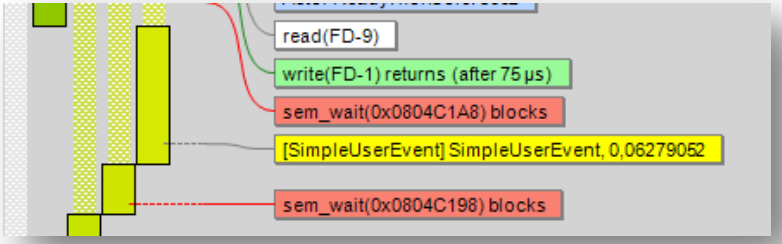
```
#include <lttng/tracepoint.h>

TRACEPOINT_EVENT(percepiodemo, simpleuserevent,
   TP_ARGS(char*, channel, float, val),
   TP_FIELDS( ctf_string(channel, channel) ctf_float(float, val, val) ) )
...
char* channel = "SimpleUserEvent";
tracepoint(percepiodemo, simpleuserevent, channel, val);
```

New easy method: tracef, a printf-style interface for LTTng tracepoint

```
#include <lttng/tracef.h>
...
tracef(“Hello world: %d", myValue);
```

New even easier method: Just write to ”/proc/lttng-logger”

```
echo -n ‘Hello world!' > /proc/lttng-logger
```



Configuring ”User Events” in Tracealyzer

Tracealyzer only displays **recognized** LTTng events. The “User Events” must therefore be specified in a ”Platform Extension” file, added in *File -> Settings*. To see ALL LTTng events, use the “Raw Trace View”.

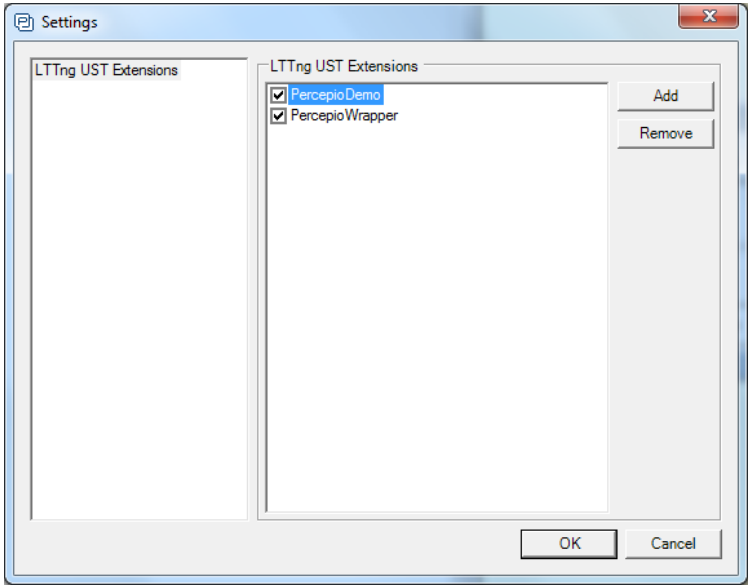
Example: PercepioDemo.xml

```
<?xml version="1.0" encoding="utf-8"?>
<PlatformExtension>
  <EventMap>
    <Event name="percepiodemo:simpleuserevent" type="UserEvent">
      <Parameter name="channel" var="channel"/>
      <Parameter name="val" var="val"/>
    </Event>
  </EventMap>
</PlatformExtension>
```

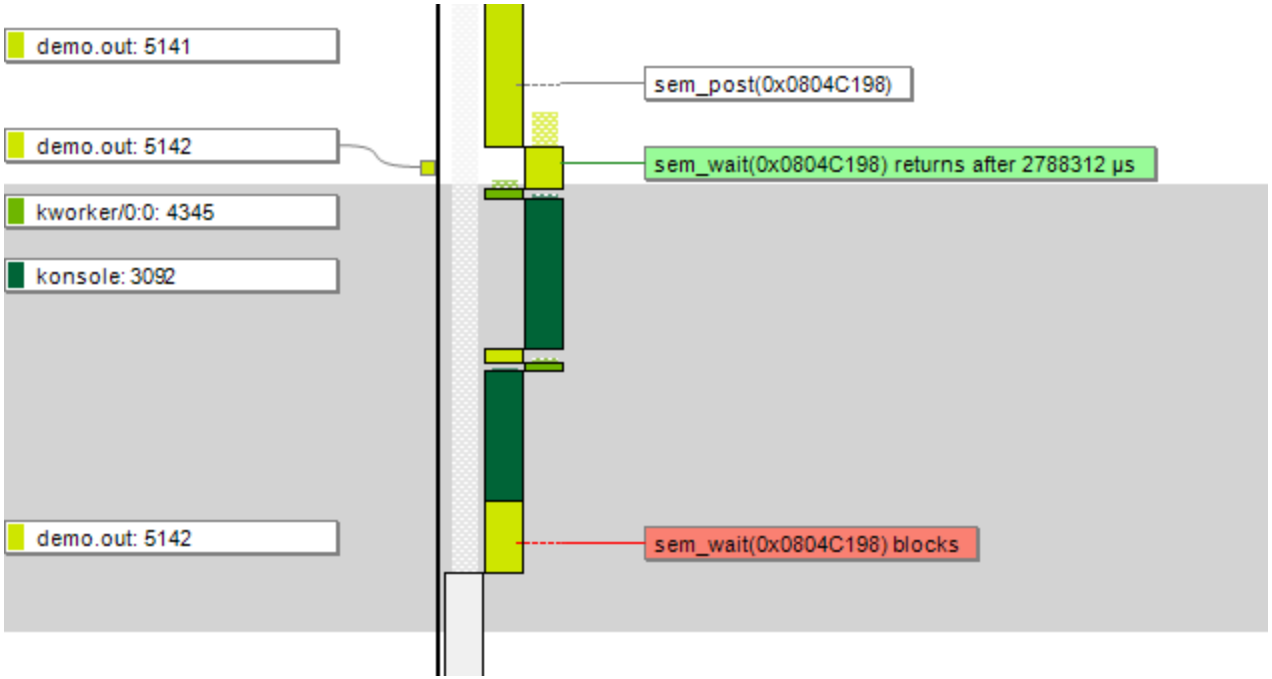
name: the Tracealyzer name for this parameter
var: the name of the LTTng tracepoint

Parameter name ”channel” has special meaning, specifies the *User Event Channel* name.

This example is available in Help menu -> ”LTTng UST Example”



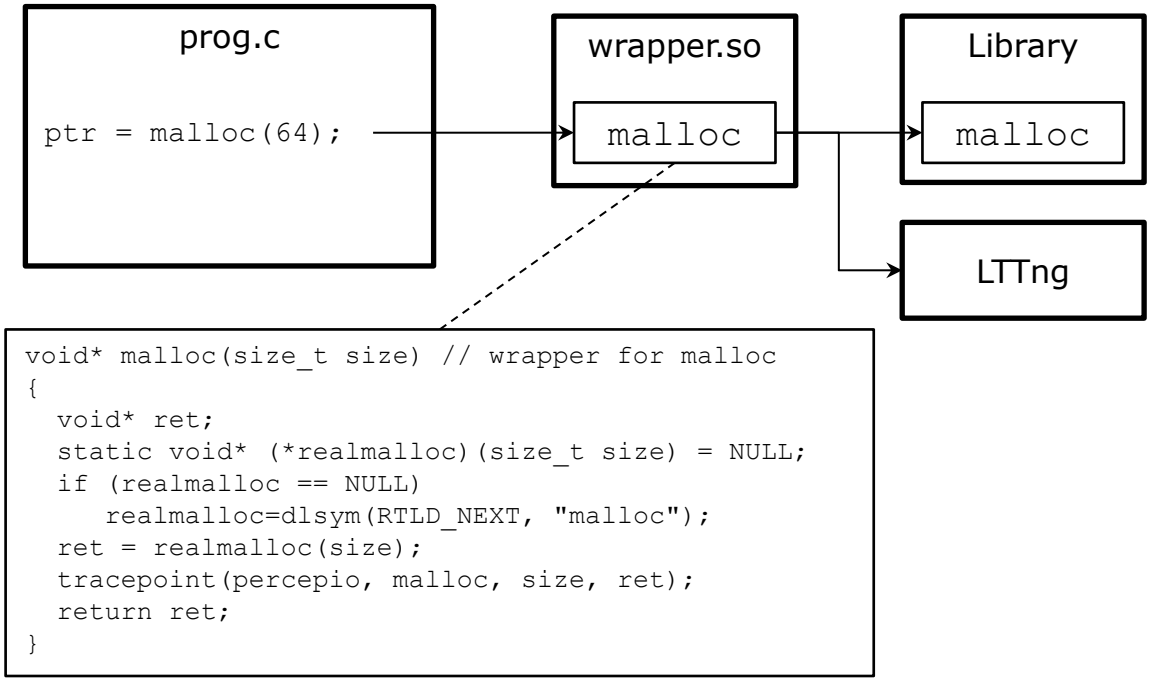
Advanced: Tracing your own ”Kernel Services”



Tracealyzer can be configured to display LTTng events from any function call as ”Kernel Services” which enables many functions like Kernel Object History and matching of related events. This requires three steps…

Step 1. Record the function calls with LTTng

```
$ LD_PRELOAD=./wrapper.so ./prog
```



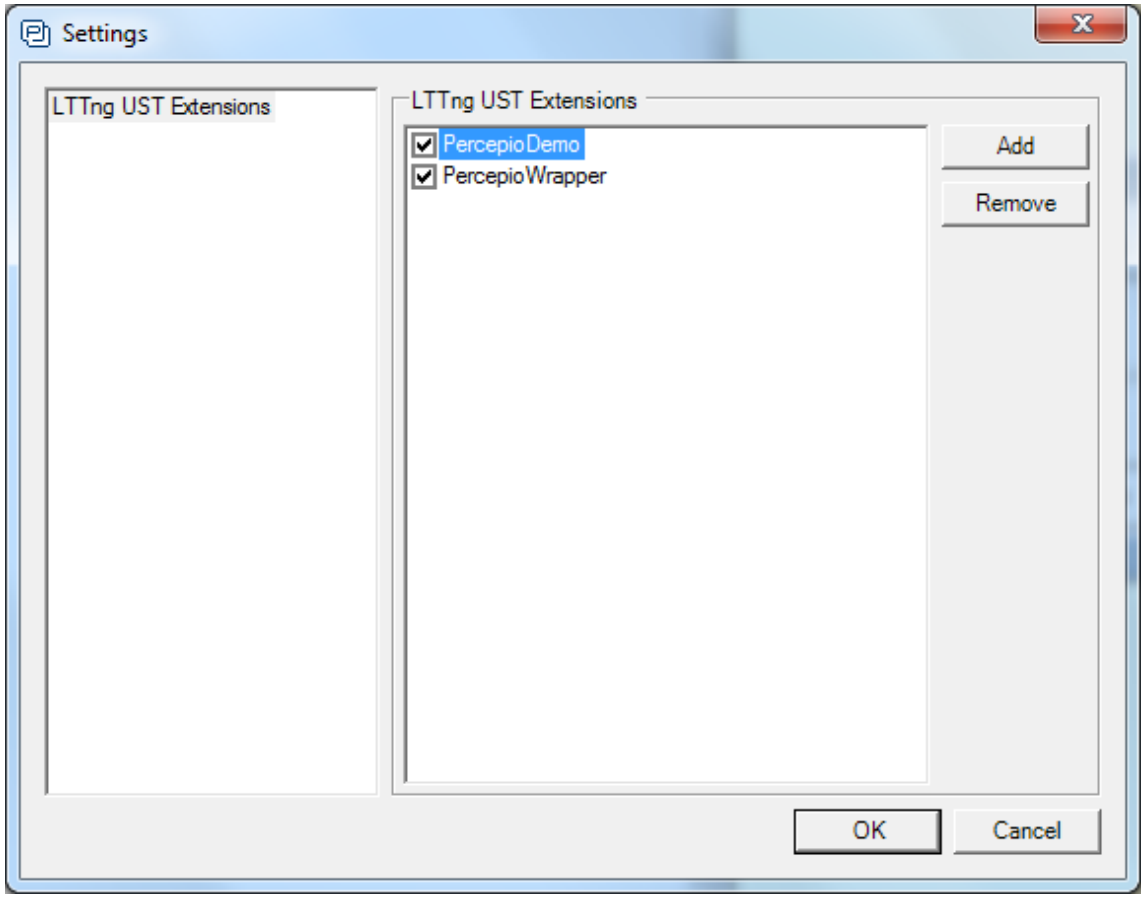
With tracepoints in wrapper functions and using LD_PRELOAD, the code don't need to be changed or even recompiled. The dynamic linking will route the calls to the new wrapper function, which calls LTTng plus the original function.

Step 2. Create a Platform Extension XML file for the new LTTng events

```
<PlatformExtension>
   <TargetPlatform>
      <KernelServiceGroups>
         <KernelServiceGroup name="UST"> - the group name showed in the View Filter
              <KernelService name="sem_wait" operation="IncreaseSemaphore"> - The name and meaning of event
                 <Parameter name="sem"/>
              </KernelService>
              ...
         </KernelServiceGroup>
      </KernelServiceGroups>
      <ObjectClasses>
          <ObjectClass name="PthreadSemaphore" type="Semaphore"/> - Type and display name of object class
       </ObjectClasses>
   </TargetPlatform>
   <ObjectMap>
      <Object class="PthreadSemaphore" format="0x{0:X8}"/> - How to format identifiers for such objects
   </ObjectMap>
   <EventMap> - Mapping between LTTng events and kernel services calls
      <Event name="percepio:sem_wait_entry" service="UST/sem_wait" type="Entry" status="Detect">
         <Parameter name="sem" var="sem" class="PthreadSemaphore"/>
      </Event>
      <Event name="percepio:sem_wait_exit" service="UST/sem_wait" type="Exit" status="Success">
         <Condition cvar="ret" cval="0" cop="&lt;" target="status" value="Timeout"/> - if (ret < 0) status = “Timeout”;
      </Event>
      ...
   </EventMap>
</PlatformExtension>
```

This example is available in Help menu -> ”LTTng UST Example”

Step 3. Import the Platform Extension



Questions?

support@percepio.com