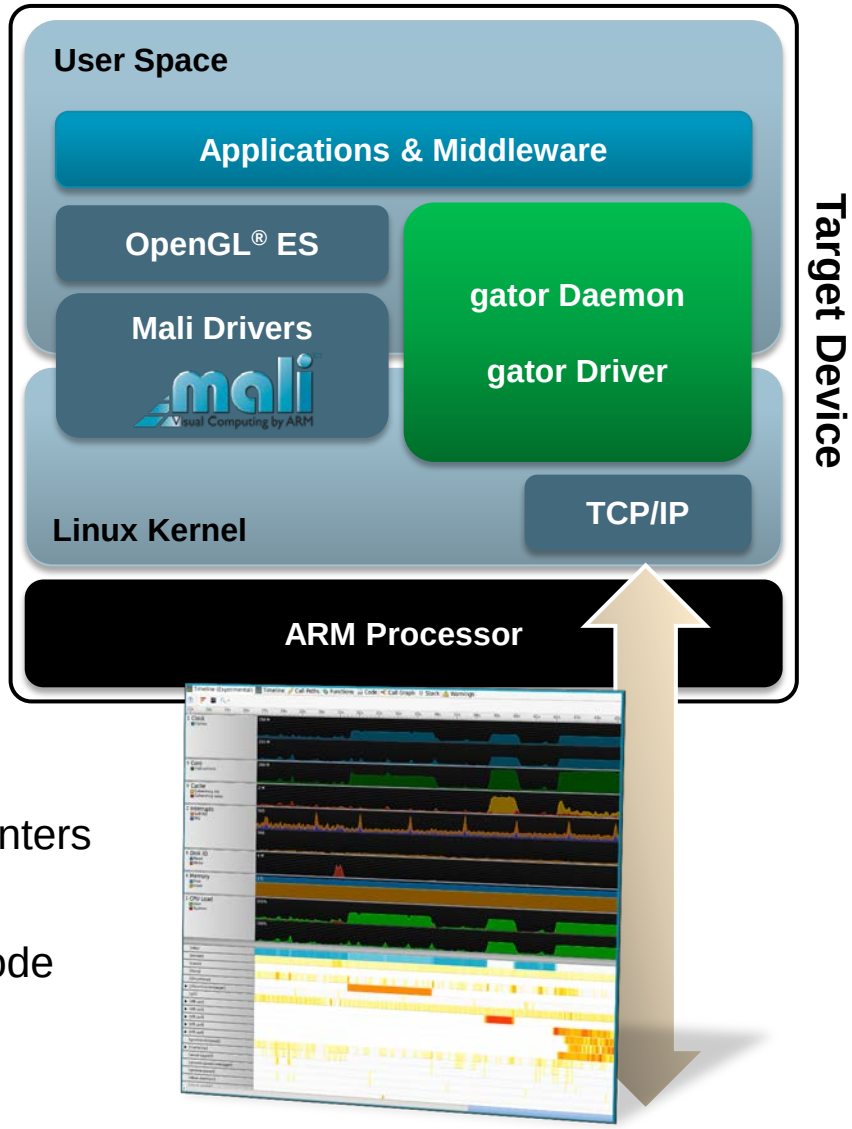
ARM Streamline and CoreSight trace

CERN Performance Tuning Workshop

22nd November 2013

Streamline basics

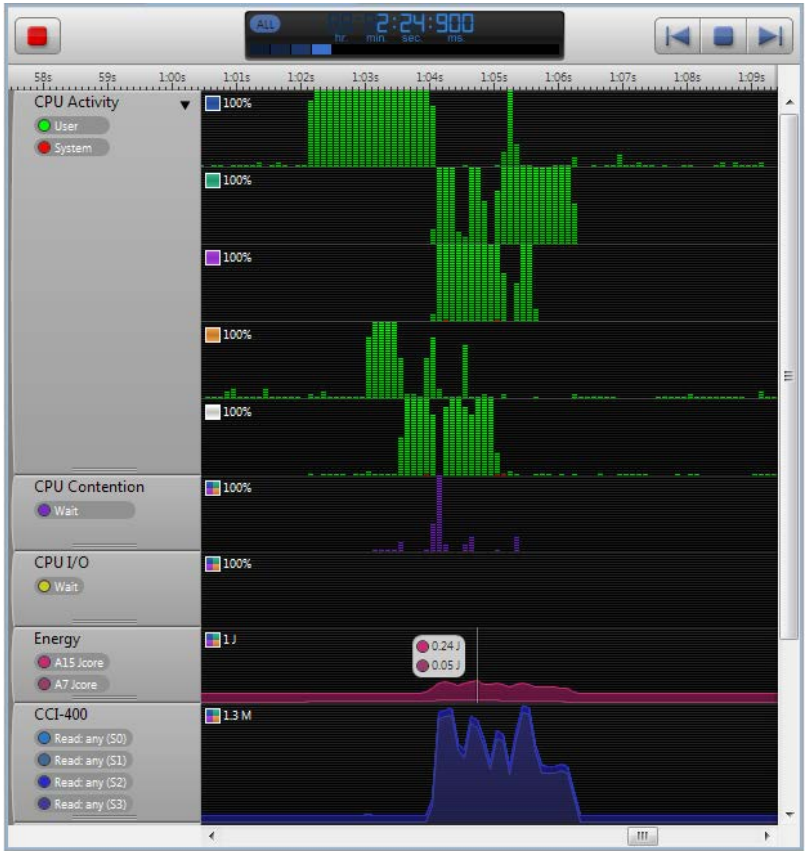
- Software based solution
  - ICE/trace units not required
  - Support for Linux kernel 2.6.32+ on target
  - Eclipse plug-in or command line
- Lightweight sample profiling
  - Time- or event*-based sampling
  - Process to C/C++ source code profiler
  - Low probe effect; <5% typically
- Multiple data sources
  - CPU, GPU and Interconnect hardware counters
  - Software counters and kernel tracepoints
  - User defined counters and instrumented code
  - Power/energy measurements



* Event-based sampling is available on kernels 3.0 or later

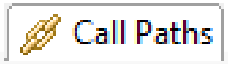
Live capture

- Charts are visible during capture
  - Great for trying to provoke an issue...
  - ....or general system monitoring
  - Think *oscilloscope*
- Watch key system metrics e.g.
  - Activity per CPU
  - Interconnect performance
  - External energy measurements
- Keep only relevant part of capture
  - Discard majority of captured data
  - Smaller files, better focus

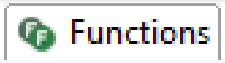


Analysis Overview

Visualization of system performance, software profile and thread switching over time

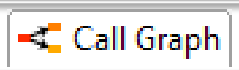


Hierarchical profile table, aggregating samples per process, thread, and function call chain

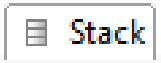


Flat software profile table, listing shared libraries and function hotspots

Source and instruction level profile. Colour coded source code lines matching samples.



Dynamically created map of the functions in your application and their relationship

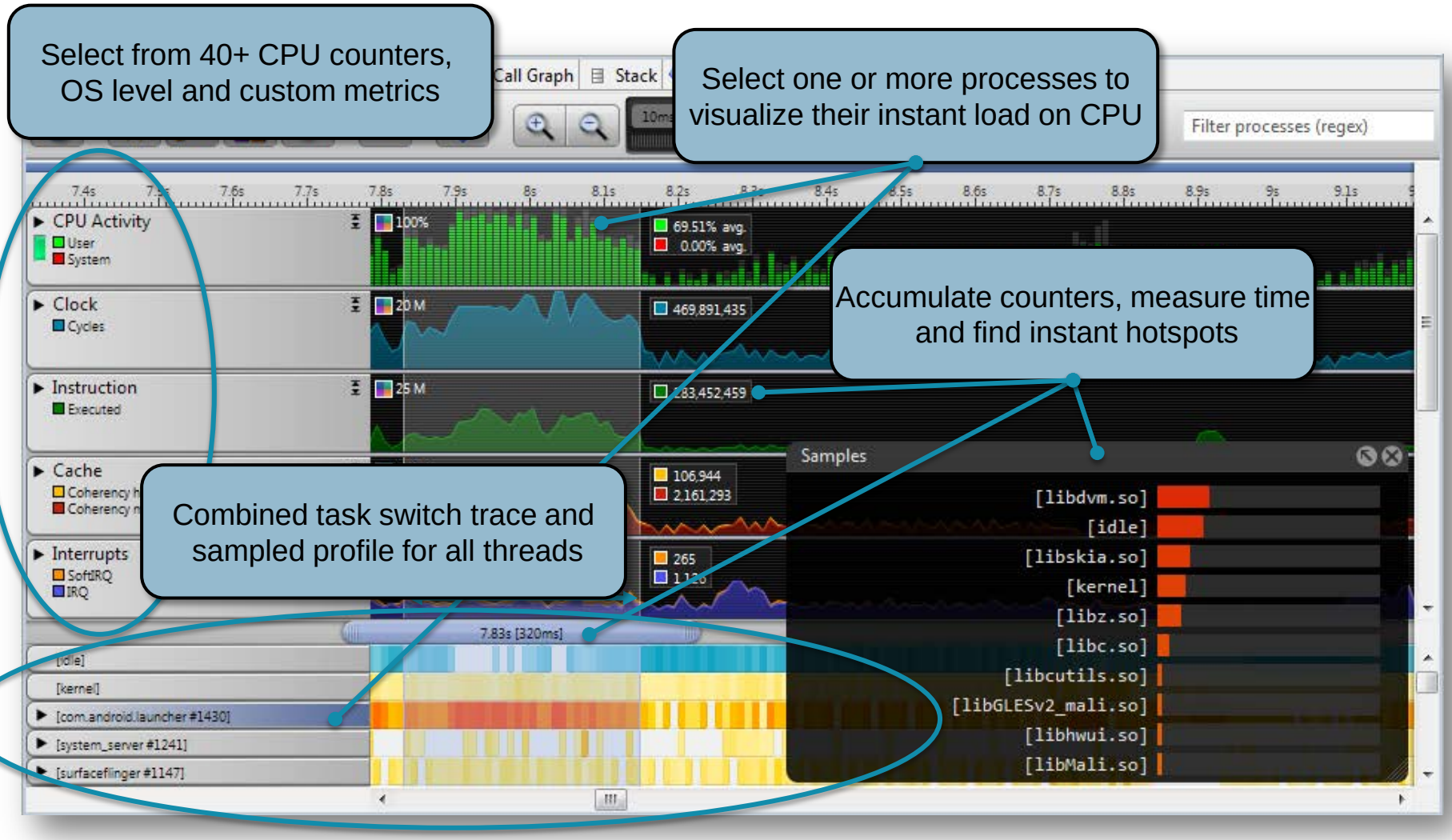


Dynamic analysis of the stack usage by your application

Chronological list of text and graphic annotations sent to target agent

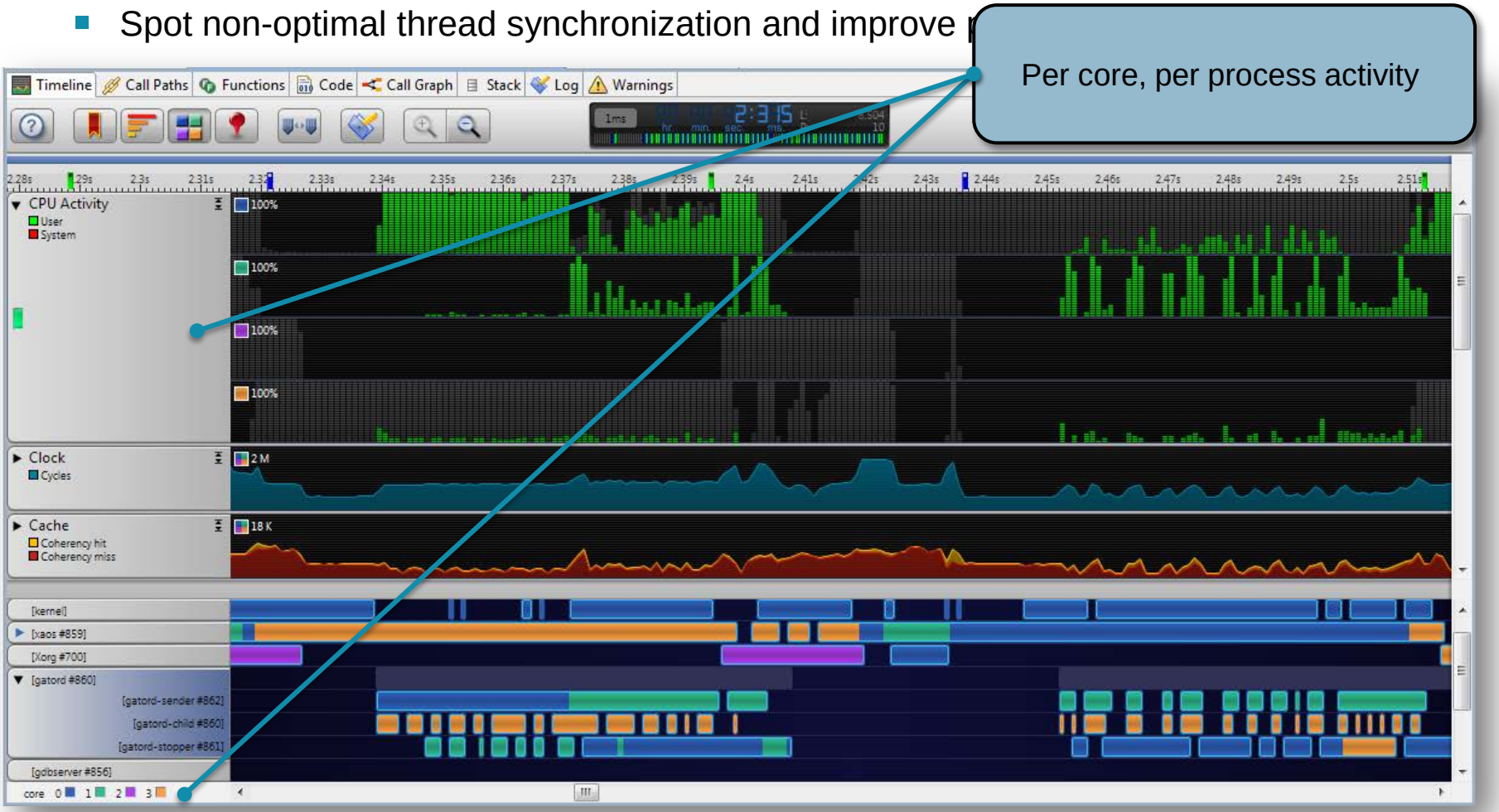
Timeline: The Big Picture

- Find hotspots, system glitches, critical conditions at a glance



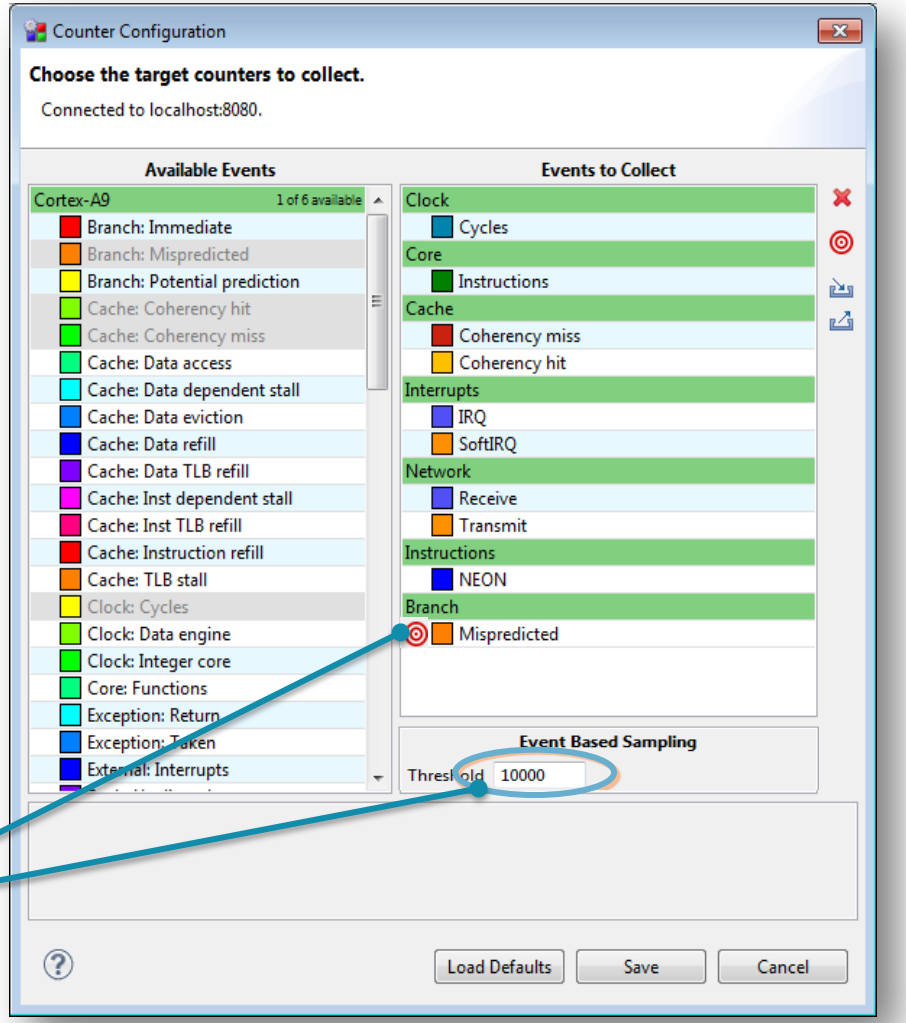
SMP Analysis

- Take advantage of multicore SMP platforms
  - Visually trace core migration and per-core statistics
  - Spot non-optimal thread synchronization and improve p



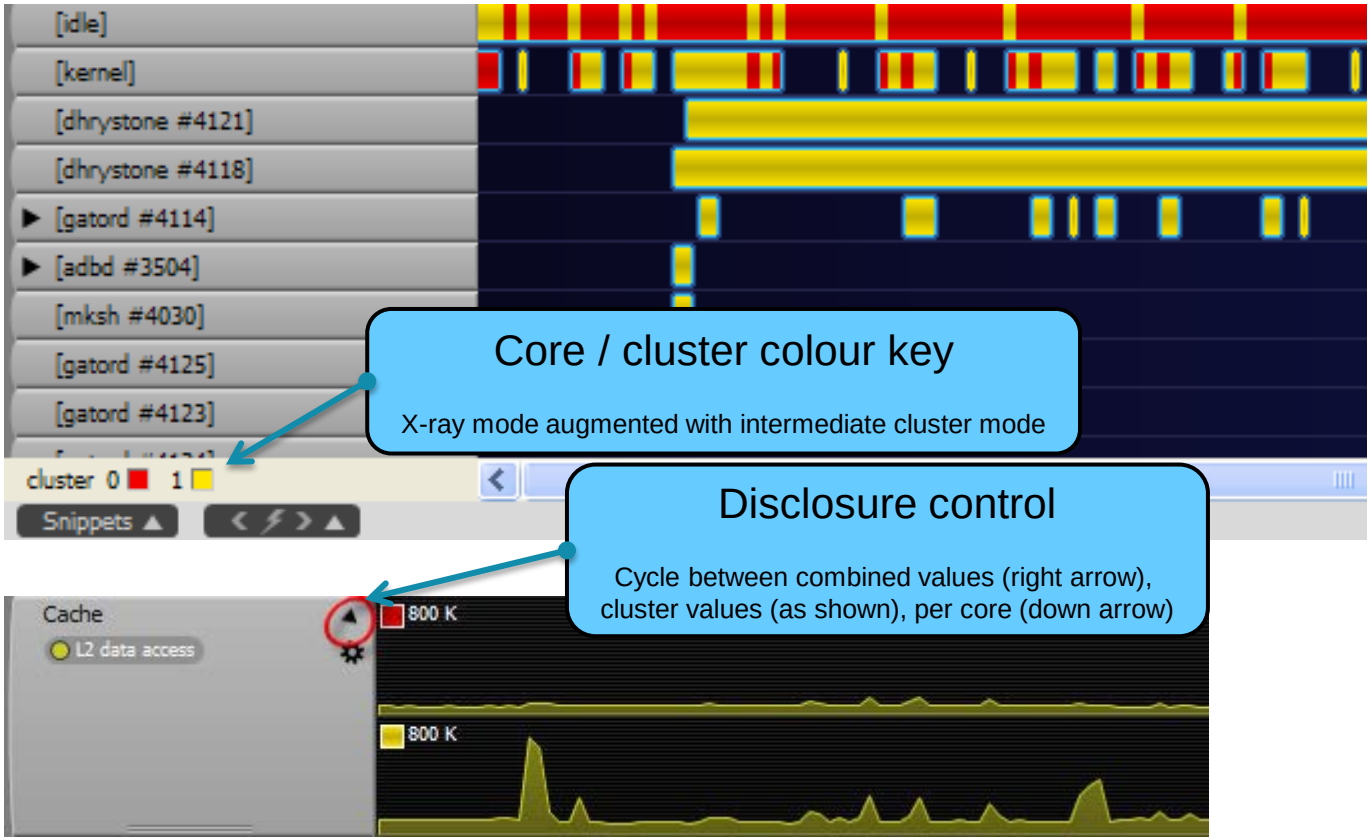
Selecting Performance Charts

- CPU aware PMU registers
  - 40+ core-level metrics to choose from
- Mali graphics
  - 300+ hardware and software counters
- OS level statistics
  - i.e. CPU load, interrupts, networking
- Custom counters
  - Easily add custom system counters
- Event-based sampling
  - Match PMU events to threads/source code



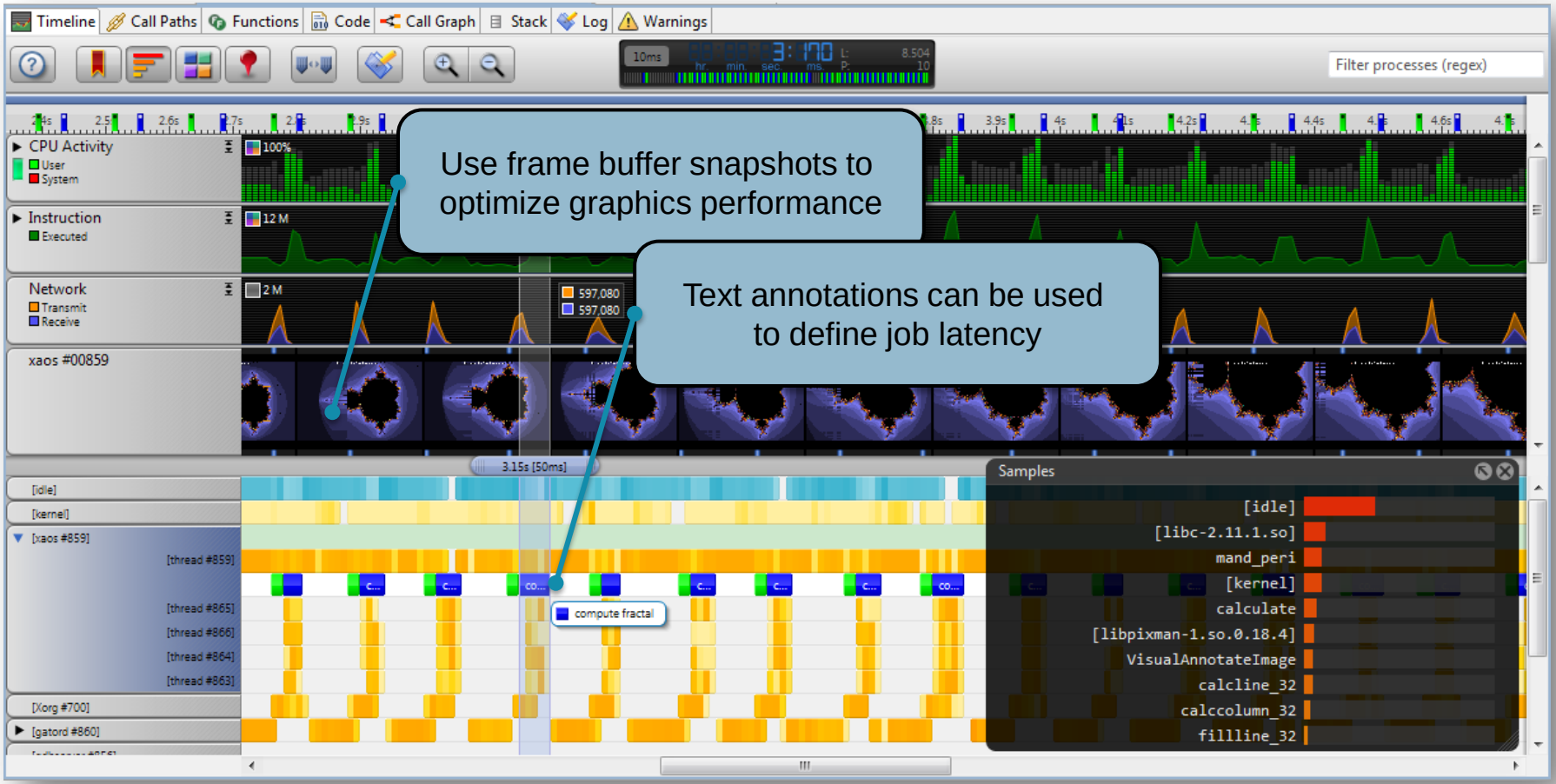
big.LITTLE Analysis

- Inspect tasks moving between clusters
  - Cycle between aggregate, per cluster and per core
  - Consistent colouring between threads and counter charts
- X-ray view
- Counters



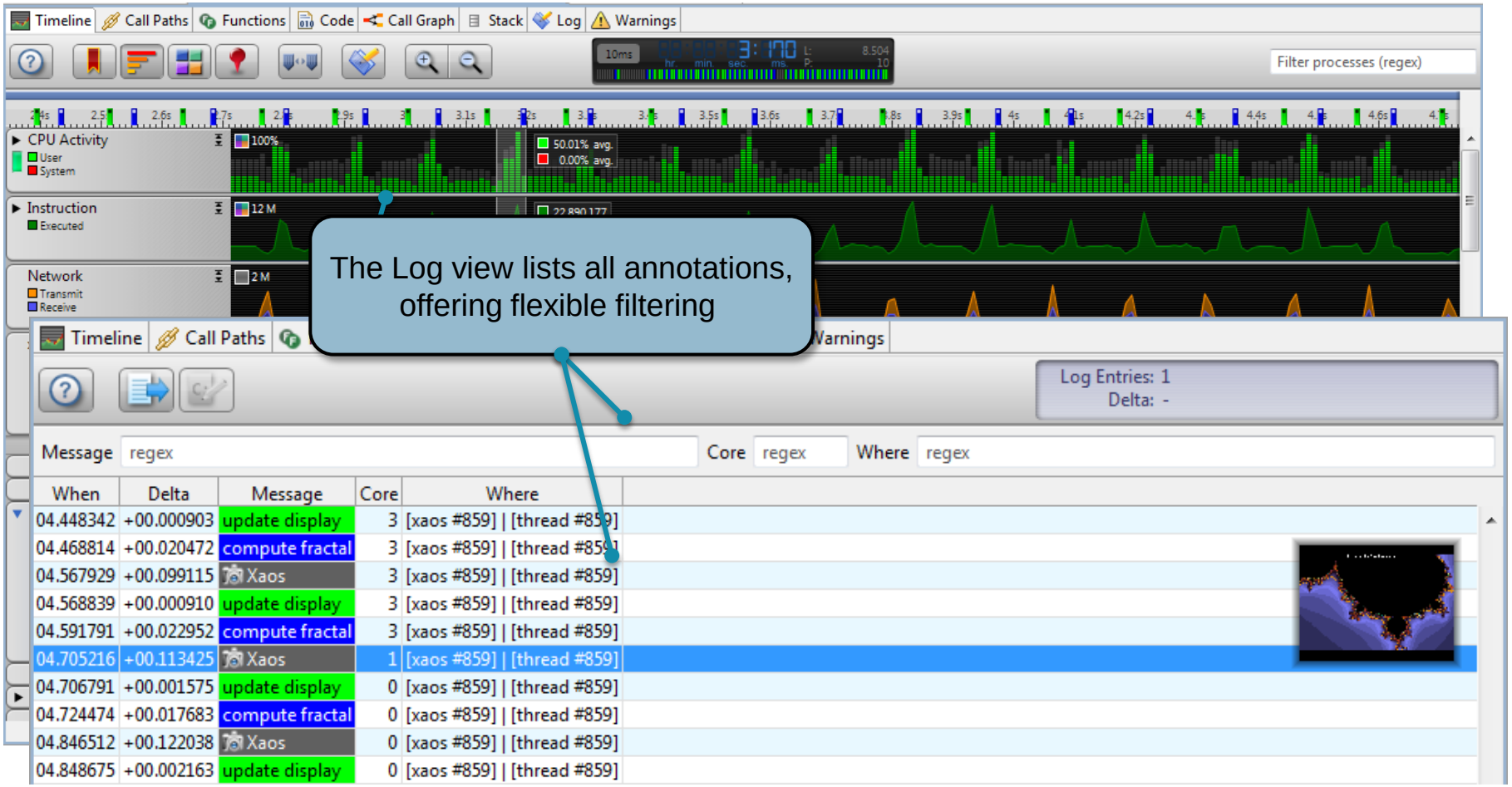
Code Instrumentation

- Output text, graphics, or markers from user or kernel space
  - Write into gator driver to get your annotations synchronized on the timeline



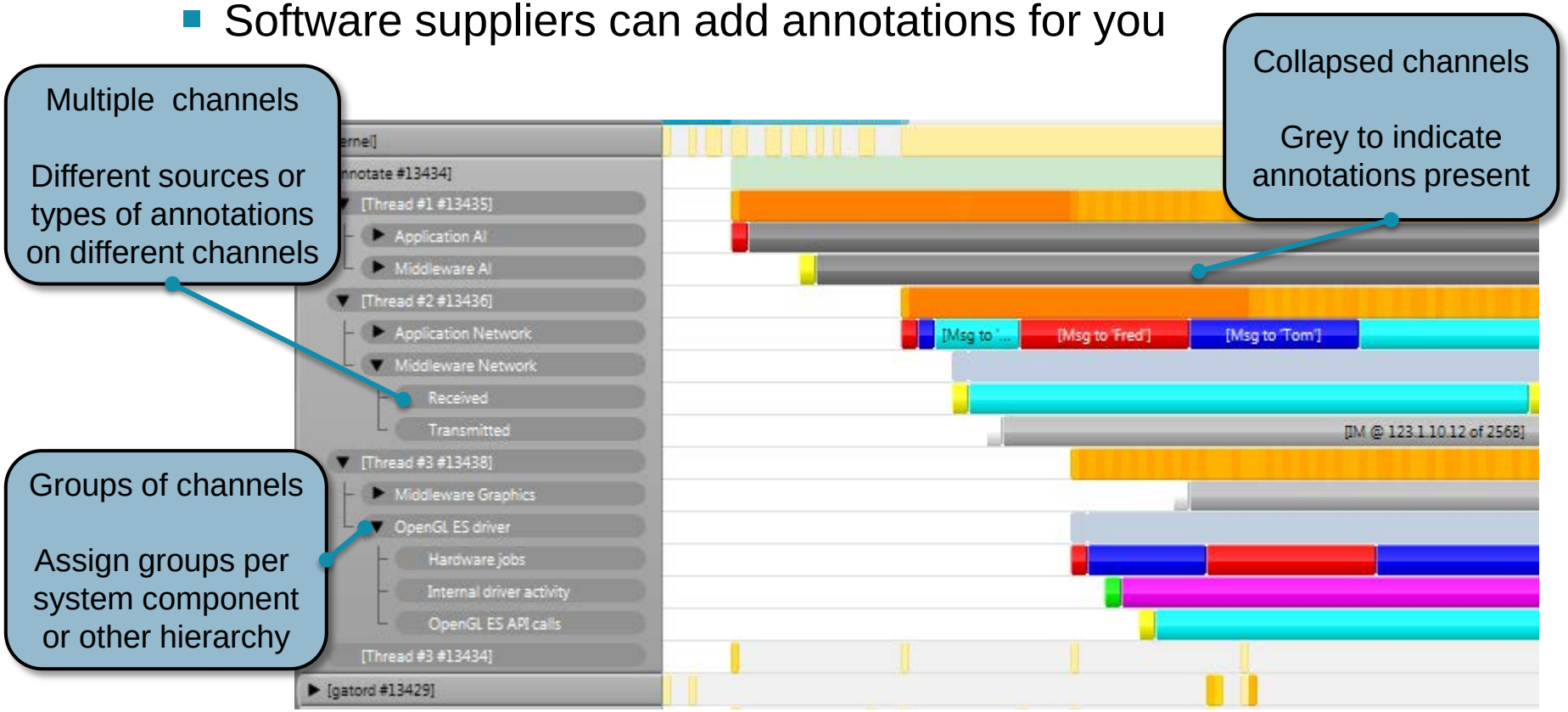
Code Instrumentation

- Output text, graphics, or markers from user or kernel space
  - Write into gator driver to get your annotations synchronized on the timeline



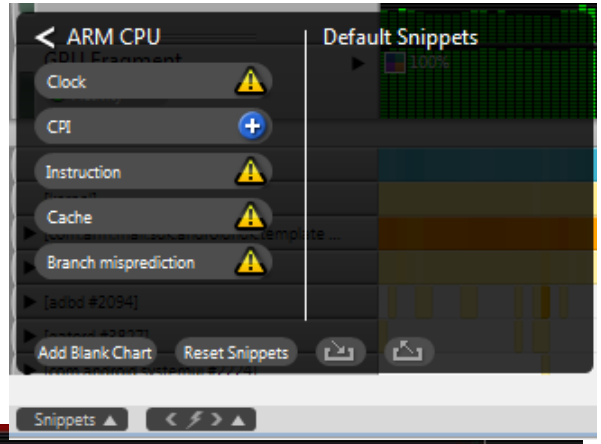
Code Instrumentation

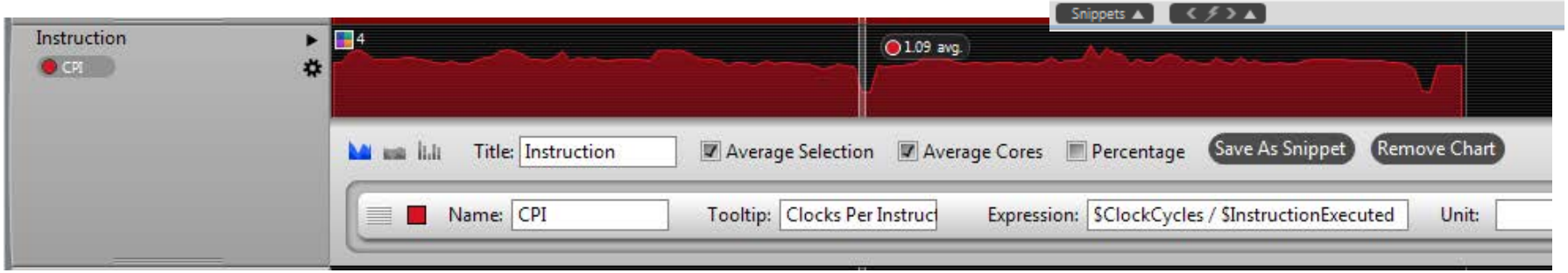
- Multiple channels and groups per thread
  - Freedom to have overlapping annotations
  - Multiple independent annotation sources (app., middleware, drivers)
    - Software suppliers can add annotations for you



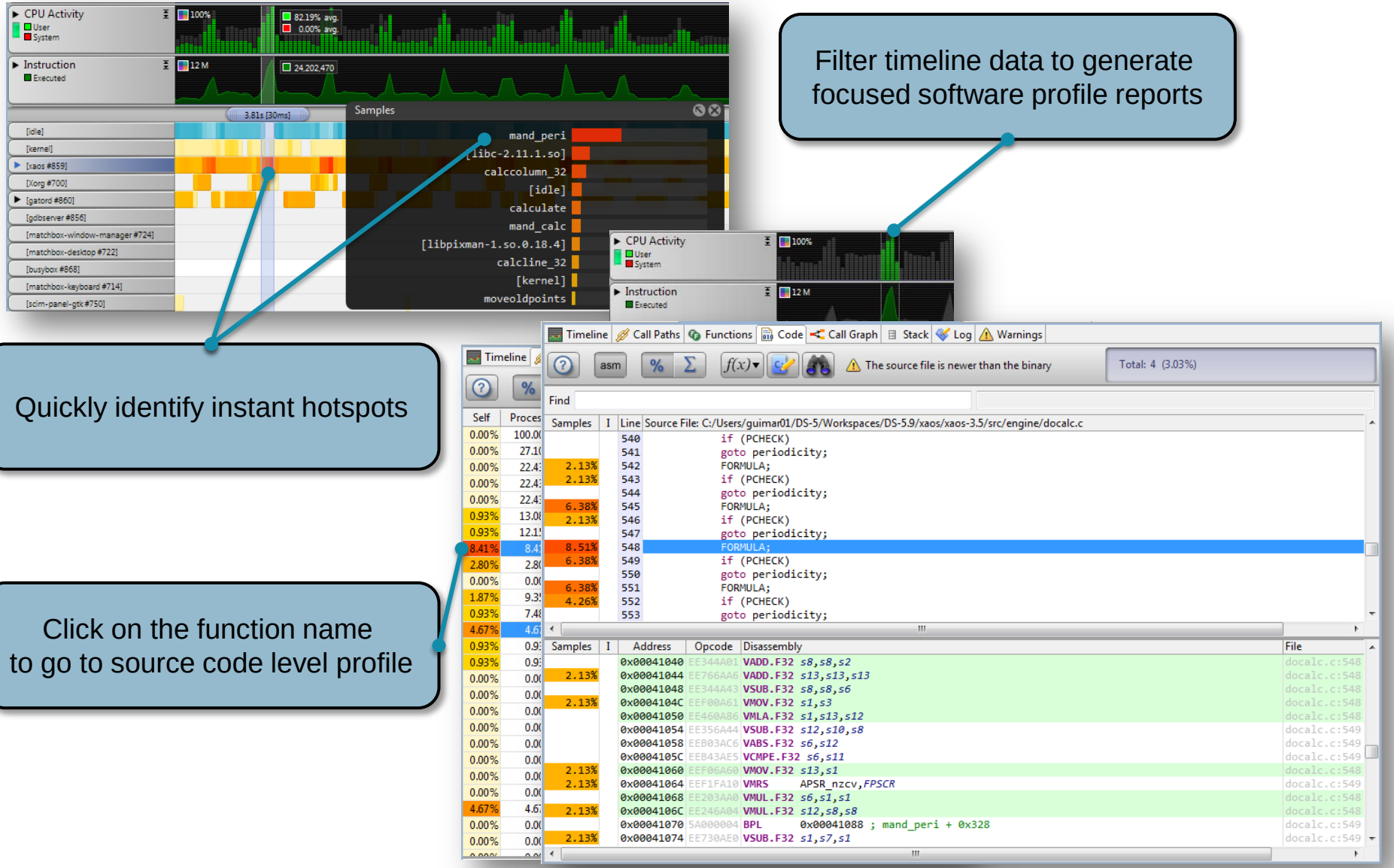
Timeline Chart Configuration

- Use expressions to create custom timeline charts
- Save, categorise and share standard expressions as ‘Snippets’





Drilldown Software Profiling

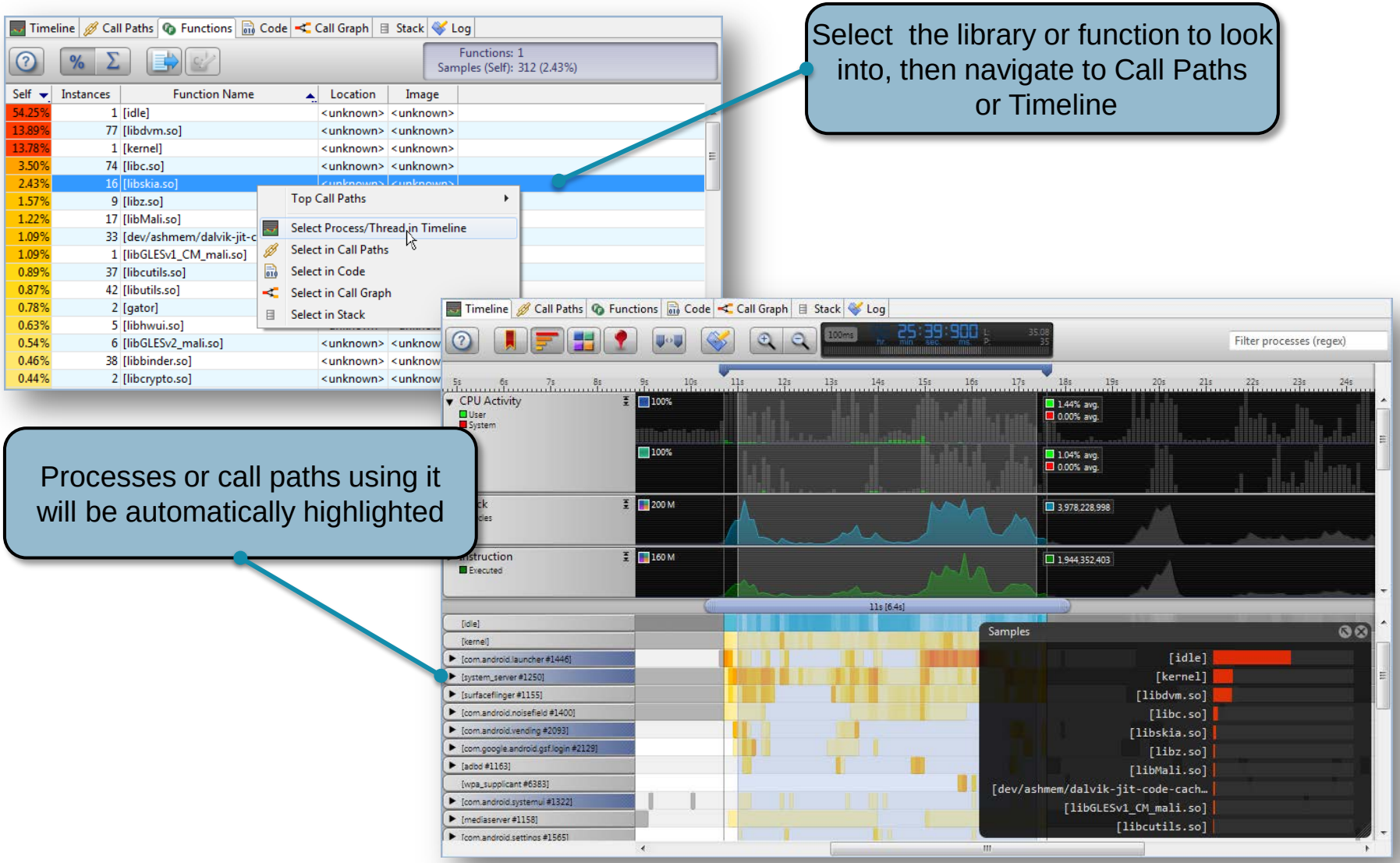


Filter timeline data to generate focused software profile reports

Quickly identify instant hotspots

Click on the function name to go to source code level profile

Bottom-Up Shared Library Analysis

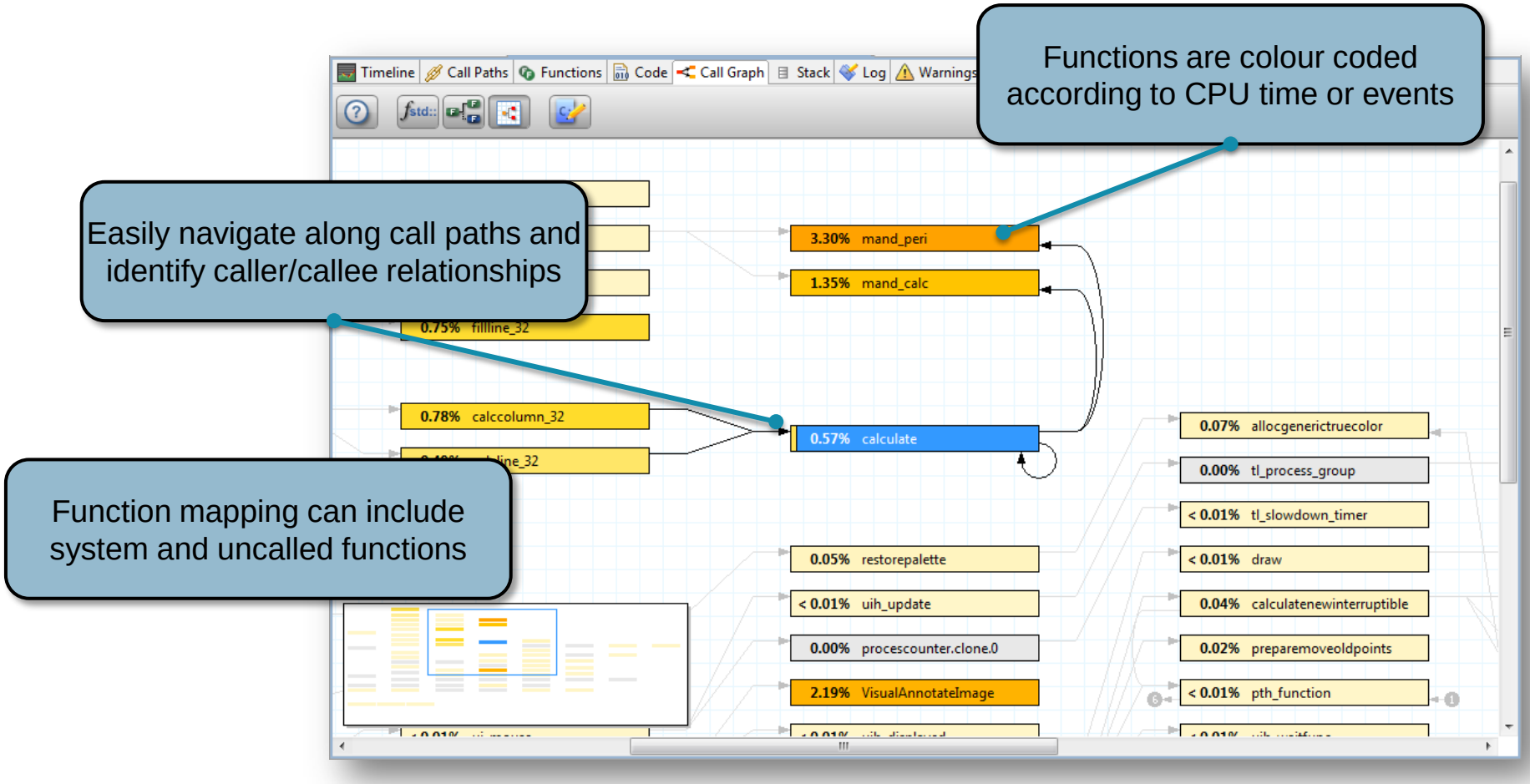


Select the library or function to look into, then navigate to Call Paths or Timeline

Processes or call paths using it will be automatically highlighted

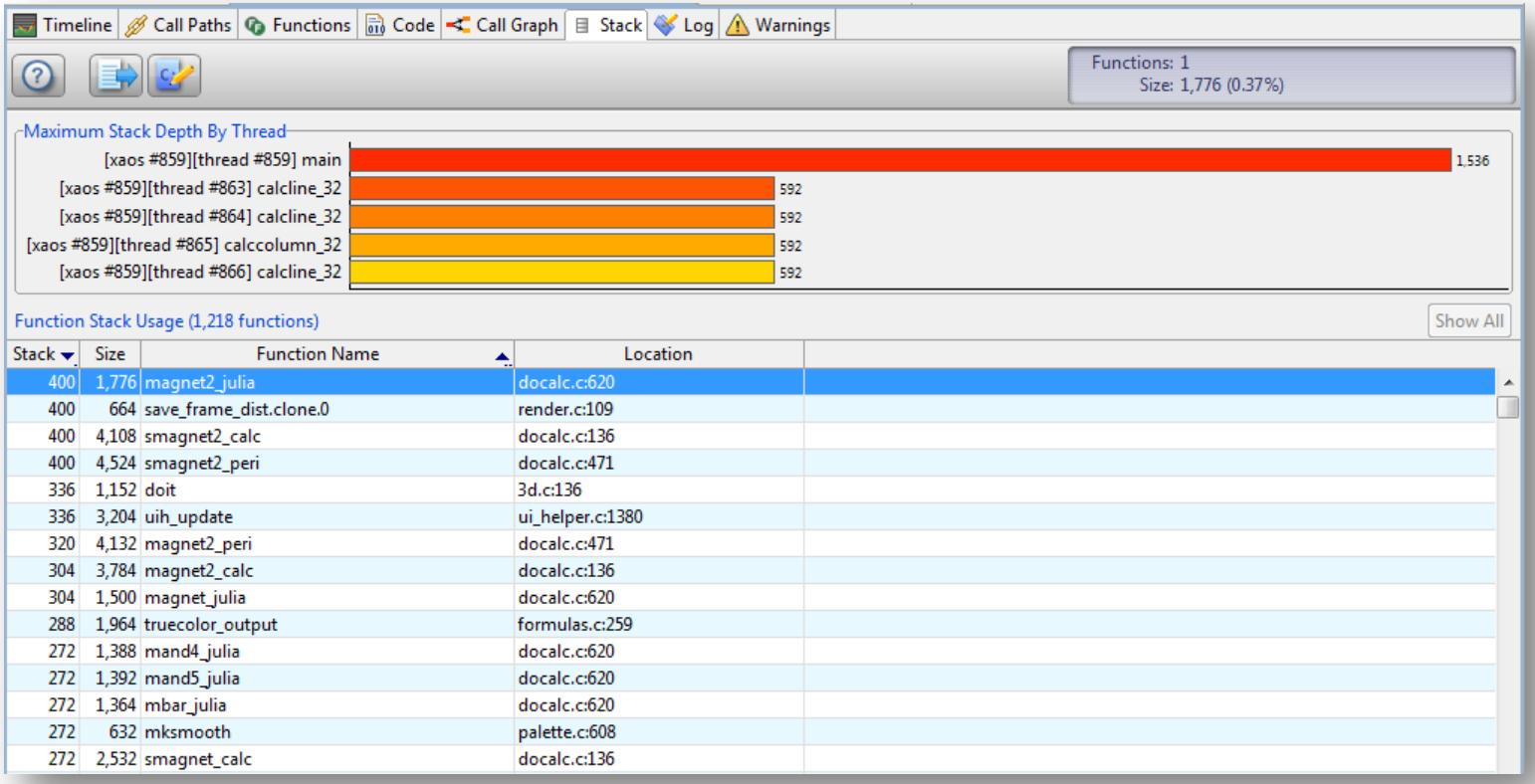
Dynamic Call Graph Analysis

- Call Graph view maps relationships between functions
  - Easy to navigate dynamic function-level map



Stack Analysis

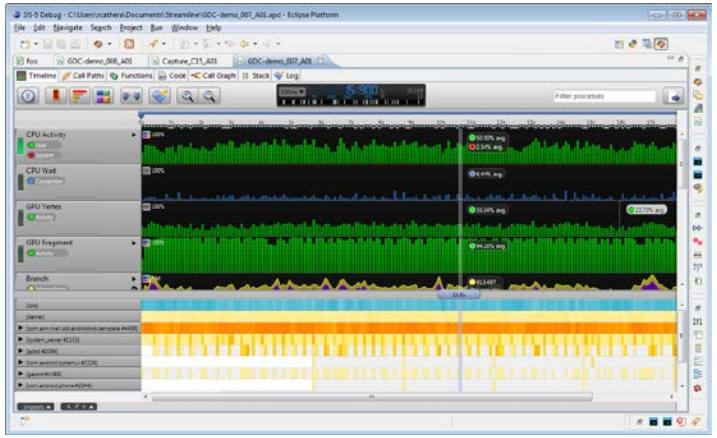
- Information on stack usage
  - Check dynamic memory usage per function



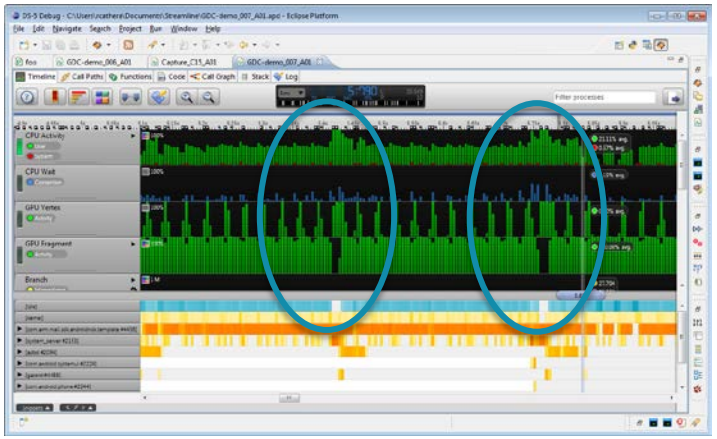
Stack	Size	Function Name	Location
400	1,776	magnet2_julia	docalc.c:620
400	664	save_frame_dist.clone.0	render.c:109
400	4,108	smagnet2_calc	docalc.c:136
400	4,524	smagnet2_peri	docalc.c:471
336	1,152	doit	3d.c:136
336	3,204	uih_update	ui_helper.c:1380
320	4,132	magnet2_peri	docalc.c:471
304	3,784	magnet2_calc	docalc.c:136
304	1,500	magnet_julia	docalc.c:620
288	1,964	truecolor_output	formulas.c:259
272	1,388	mand4_julia	docalc.c:620
272	1,392	mand5_julia	docalc.c:620
272	1,364	mbar_julia	docalc.c:620
272	632	mksmooth	palette.c:608
272	2,532	smagnet_calc	docalc.c:136

Beyond average...

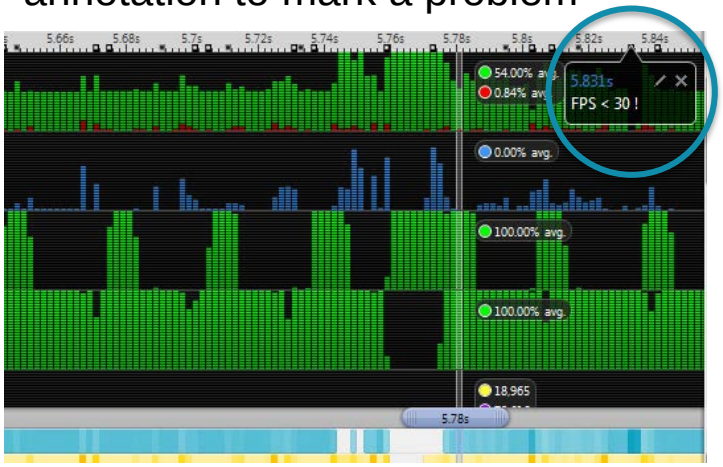
1. Average looks okay but we know we have glitches



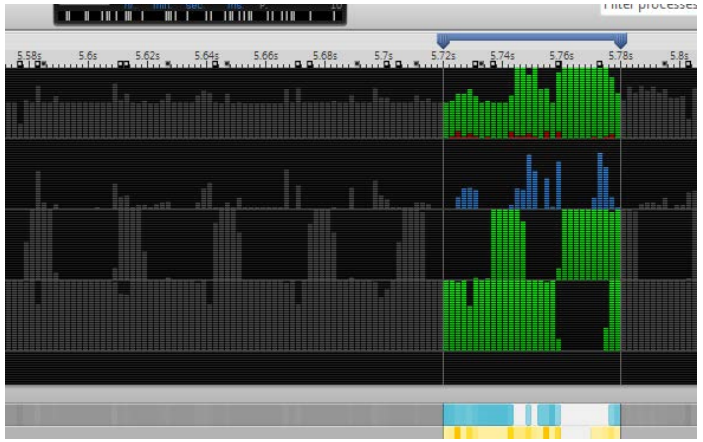
2. Zooming in we can quickly identify something different...



3. ...or we can write out an annotation to mark a problem



4. Then isolate region of interest with callipers and re-compute statistics



Power Measurement Interfaces

Data Acquisition

ARM Energy Probe

- 3-channel
- System-level analysis
- Easy to deploy
- Affordable

Good for trend spotting and application optimization

NI DAQ USB-62xx

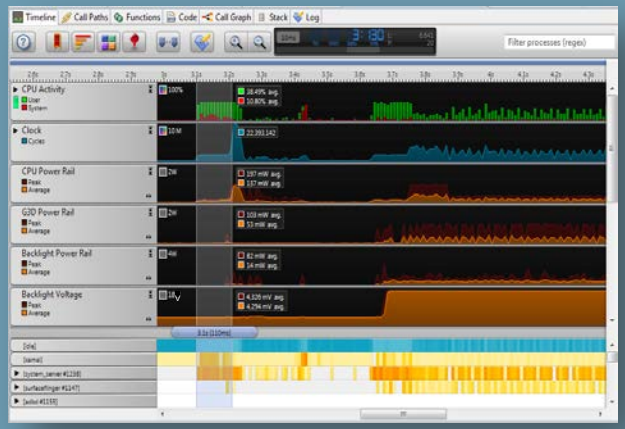
- 40+ analog inputs
- Subcomponent sensitivity
- High fidelity
- Higher cost

Good for OS power management tuning and benchmarking

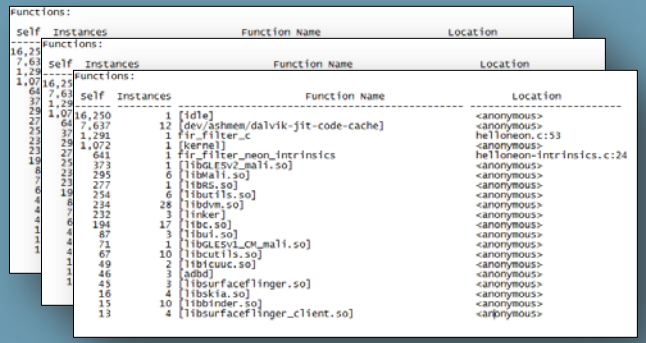
also: on-chip sensors (via hwmon)

Streamline

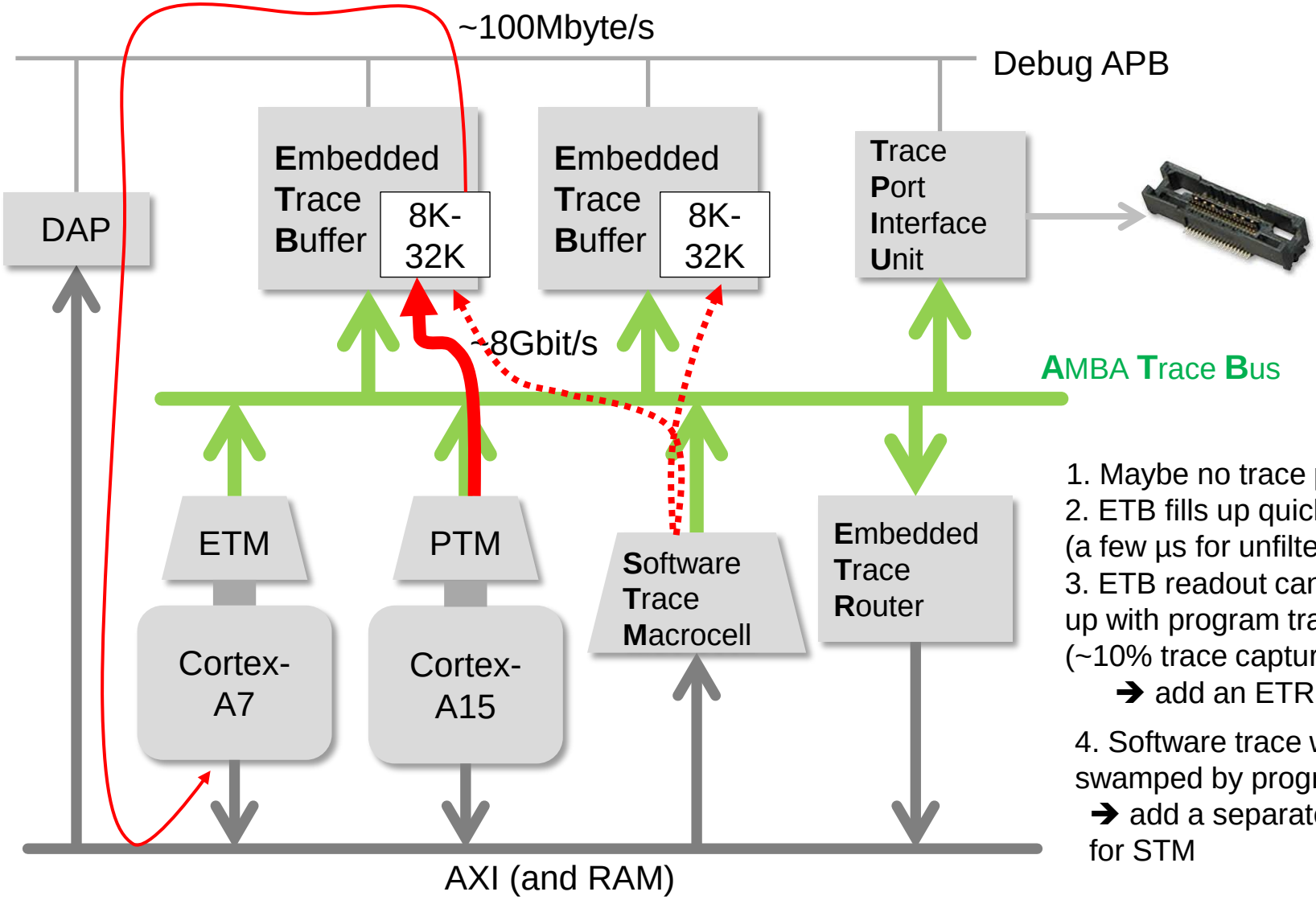
Visual Analysis



Automated Tests



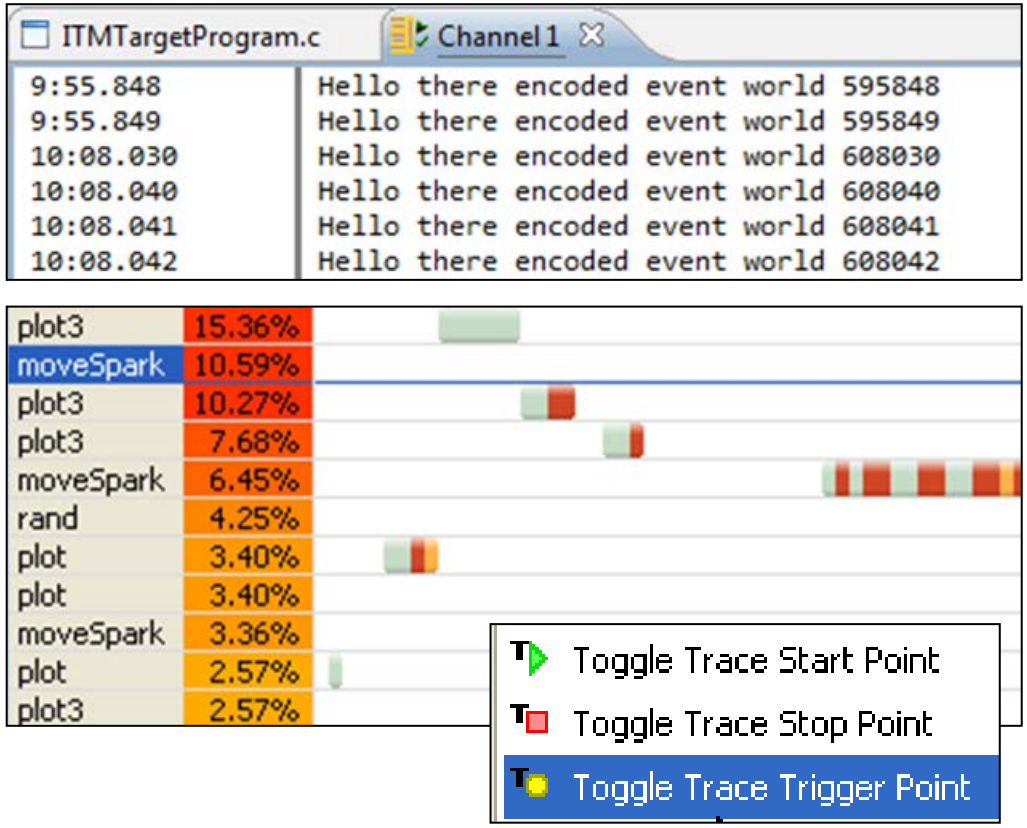
CoreSight trace



1. Maybe no trace port
2. ETB fills up quickly
(a few µs for unfiltered trace)
3. ETB readout can't keep up with program trace
(~10% trace captured)
➔ add an ETR
4. Software trace will be swamped by program trace
➔ add a separate ETB for STM

Profiling with CoreSight Trace

Trace view in DS-5 Debugger



- ITM/STM Event Viewer to track software execution
- ETM/PTM instruction and data trace to pin-point software bugs
- Tracepoints and filters to optimize the usage of on-chip trace buffers
- Instruction trace based profiling reports

Cycle-based tracing

```
 31 ] (cpu2) --> vector_swi
 86 ] (cpu2) --> trace_hardirqs_on
163 ] (cpu2) --> mark_held_locks
199 ] (cpu2) --> mark_held_locks
261 ] (cpu2) --> sys_getppid
274 ] (cpu2) --> lock_acquire
318 ] (cpu2) --> __lock_acquire
451 ] (cpu2) --> mark_lock
605 ] (cpu2) --> pid_vnr
638 ] (cpu2) --> lock_release
765 ] (cpu2) --> ret_fast_syscall
770 ] (cpu2) --> trace_hardirqs_off
```

core #1: 739 cycles

```
 96 ] (cpu1) --> vector_swi
299 ] (cpu1) --> trace_hardirqs_on
350 ] (cpu1) --> mark_held_locks
356 ] (cpu1) --> mark_held_locks
433 ] (cpu1) --> sys_getppid
438 ] (cpu1) --> lock_acquire
529 ] (cpu1) --> __lock_acquire
638 ] (cpu1) --> mark_lock
760 ] (cpu1) --> pid_vnr
773 ] (cpu1) --> lock_release
948 ] (cpu1) --> ret_fast_syscall
1004 ] (cpu1) --> trace_hardirqs_off
```

core #2: 908 cycles

Non-invasive and works with IRQs disabled

Self-hosted debug/trace

- Self-hosted trace control library releasing end 2013
- Targeted tracing for performance investigations
  - enable/disable trace round region of interest
- Sampling profiler/coverage tool
  - repeatedly capture trace fragments (cycle-accurate)
  - allow accurate measurement of basic block execution times
  - can use “shotgun sequencing” to construct a global profile
- Target-resident self-test
  - quickly and systematically/randomly iterate through multiple configs
- System “flight recorder”
  - capture rolling trace into ETB from boot time onwards
  - stop capture when fault detected
    - can use cross-triggers from CPU, system etc. to stop trace
  - create crash dump including ETB contents
  - SiP/OEM want this for base stations, network processors etc.

Performance analysis - where next?

- More structure to annotations
  - define start and end of interval
  - associate resource usage with intervals
- Scale to multiple devices and clusters
- Improved support for GPGPU (OpenCL)
- Closer integration of processor trace and sample-based profiling
- Use CoreSight STM for trace/profile transport
- Standardize Linux interfaces to on-target trace
  - enable/disable trace on perf events
  - sideband data for trace decompression

END

www.arm.com/streamline

BACKUP

Streamline Community vs. Basic/Pro

- Which is the right Streamline for you?

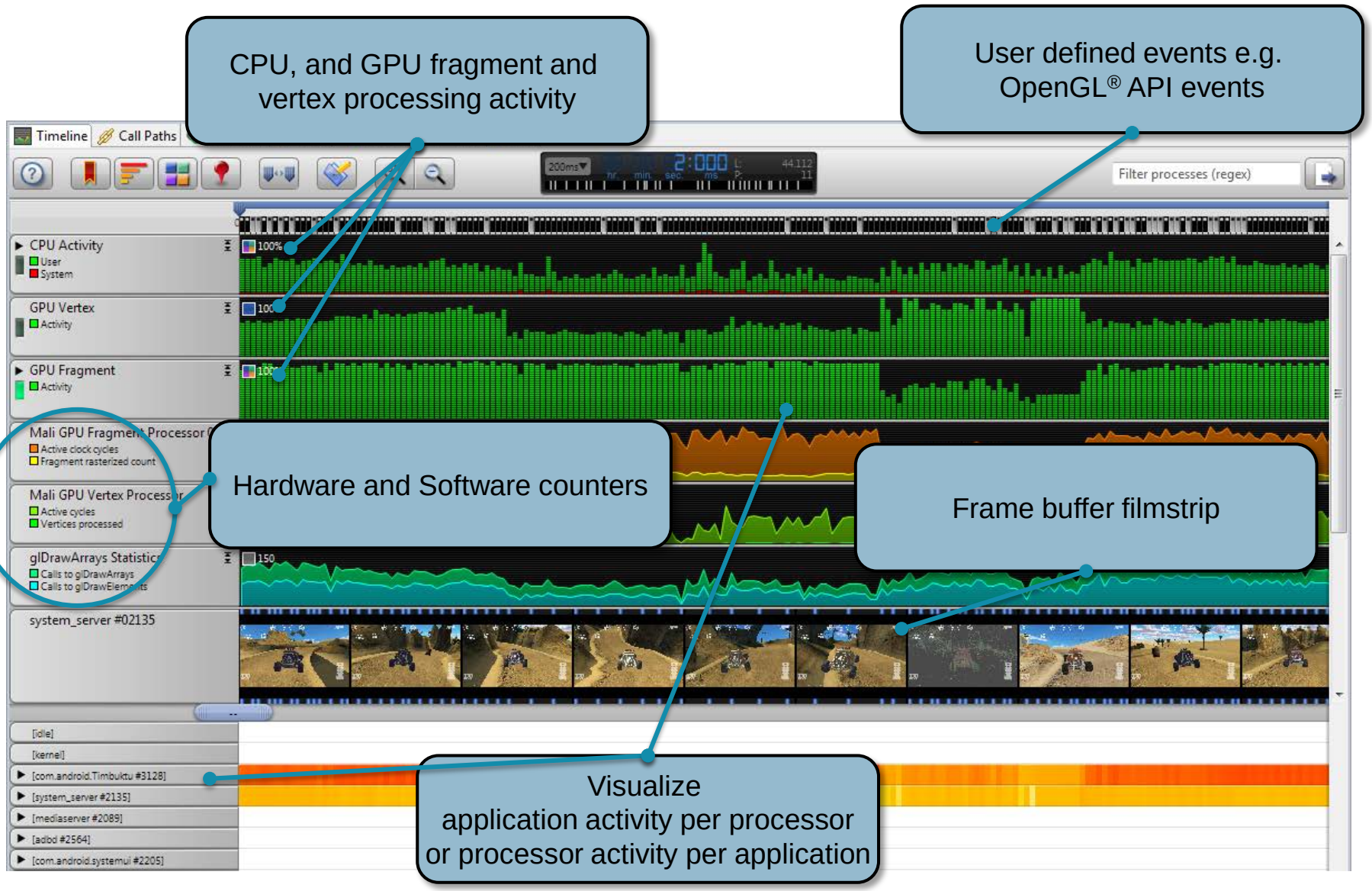
	Community	Basic/Pro
Typical Use Case	Simple application profiling	System-wide, SMP analysis
Program Images	1	Limited to host memory
Timeline View		
*** Performance Charts**	✓	✓
*** Process Bars**	✓	✓
*** Mali GPU Analysis**	✓	✓
*** Quick Profile Summary**		✓
*** Core Affinity Mode**		✓
*** Energy Probe data capture**		✓
*** Time Filtering**		✓
*** Annotation**	✓	✓
Call Paths View		✓
Functions View	✓	✓
Code View		✓
Call Graph		✓
Stack View		✓
Log View		✓
Command Line		✓
Event Based Sampling		✓

Instruction Trace or Sampling?

- Choosing the right tool for the job
  - Instruction trace-based analysis doesn't scale to high-end Cortex-A

Instruction trace-based	Streamline Sample-based
Pros	**Pros**
High granularity	Integrates system events and PMU counters
Non-intrusive	Hours+ capture time
	Can be run remotely and in production units
Cons	**Cons**
Cost and scalability - Requires trace unit and off-chip trace ports	Cannot be used to observe short instruction sequences
Sub-second capture time	Adds single digit overhead
Complex to set up	
No standard solution for dynamically loaded code	

GPU Graphics Analysis



Command Line Interface

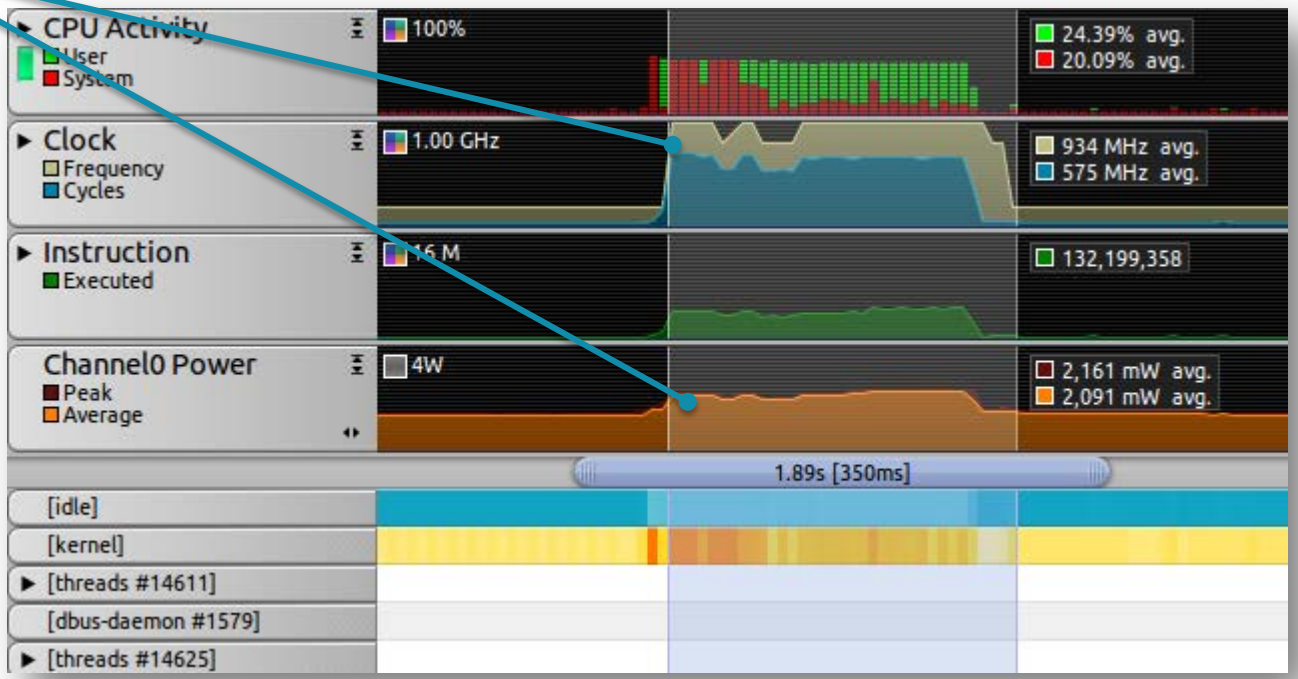
- Enables automated scripted workflows
  - Manual or timed data capture
  - Filter by runtime defined start and stop bookmarks – great for benchmarks
  - Generates text-based reports: function, call path, stack, and log views
- Parse and compare reports for testing or benchmarking

```
Functions:

 Self  Instances                    Function Name                          Location
------ --------- ------------------------------------------------ ------------------------
16,250         1 [idle]                                           <anonymous>
 7,637        12 [dev/ashmem/dalvik-jit-code-cache]              <anonymous>
 1,291         1 fir_filter_c                                    helloneon.c:53
 1,072         1 [kernel]                                        <anonymous>
   641         1 fir_filter_neon_intrinsics                      helloneon-intrinsics.c:24
   373         1 [libGLESv2_mali.so]                             <anonymous>
   295         6 [libMali.so]                                    <anonymous>
   277         1 [libRS.so]                                      <anonymous>
   254         6 [libutils.so]                                   <anonymous>
   234        28 [libdvm.so]                                     <anonymous>
   232         3 [linker]                                        <anonymous>
   194        17 [libc.so]                                       <anonymous>
    87         3 [libui.so]                                      <anonymous>
    71         1 [libGLESv1_CM_mali.so]                          <anonymous>
    67        10 [libcutils.so]                                  <anonymous>
    49         2 [libicuuc.so]                                   <anonymous>
    46         3 [adbd]                                          <anonymous>
    45         3 [libsurfaceflinger.so]
    16         4 [libskia.so]
    15        10 [libbinder.so]
    13         4 [libsurfaceflinger_client.so]
```

```
Options available to all modes:

  -h, -?, -help            Displays a description of each option
  -v, -version             Displays the program version information
  -o, -output <output>     Sends all output text into the specified file

Additional options available to report mode:

  -all                     Produce all tables [default]
  -callpath                Produce the call paths table
  -function                Produce the functions table
  -stack                   Produce the function stack usage table
  -log                     Produce the log entry table
  -format <space|tab|csv>  Format the tables using spaces, tabs or
                           comma-separated values [default: space]
```

DVFS in Practice

Track DVFS frequency scaling and its impact in power consumption



The *Power* of Having It All in One Place

- How effective are you managing your energy budget?