FTF 2016
TECHNOLOGY FORUM

OPTIMIZE PERFORMANCE OF LINUX APPLICATIONS USING CW-ARMV8

ROBERT MCGOWAN | CHIEF ARCHITECT
RAZVAN IONESCU

SESSION #FTF-DES-N1835
19, MAY, 2016

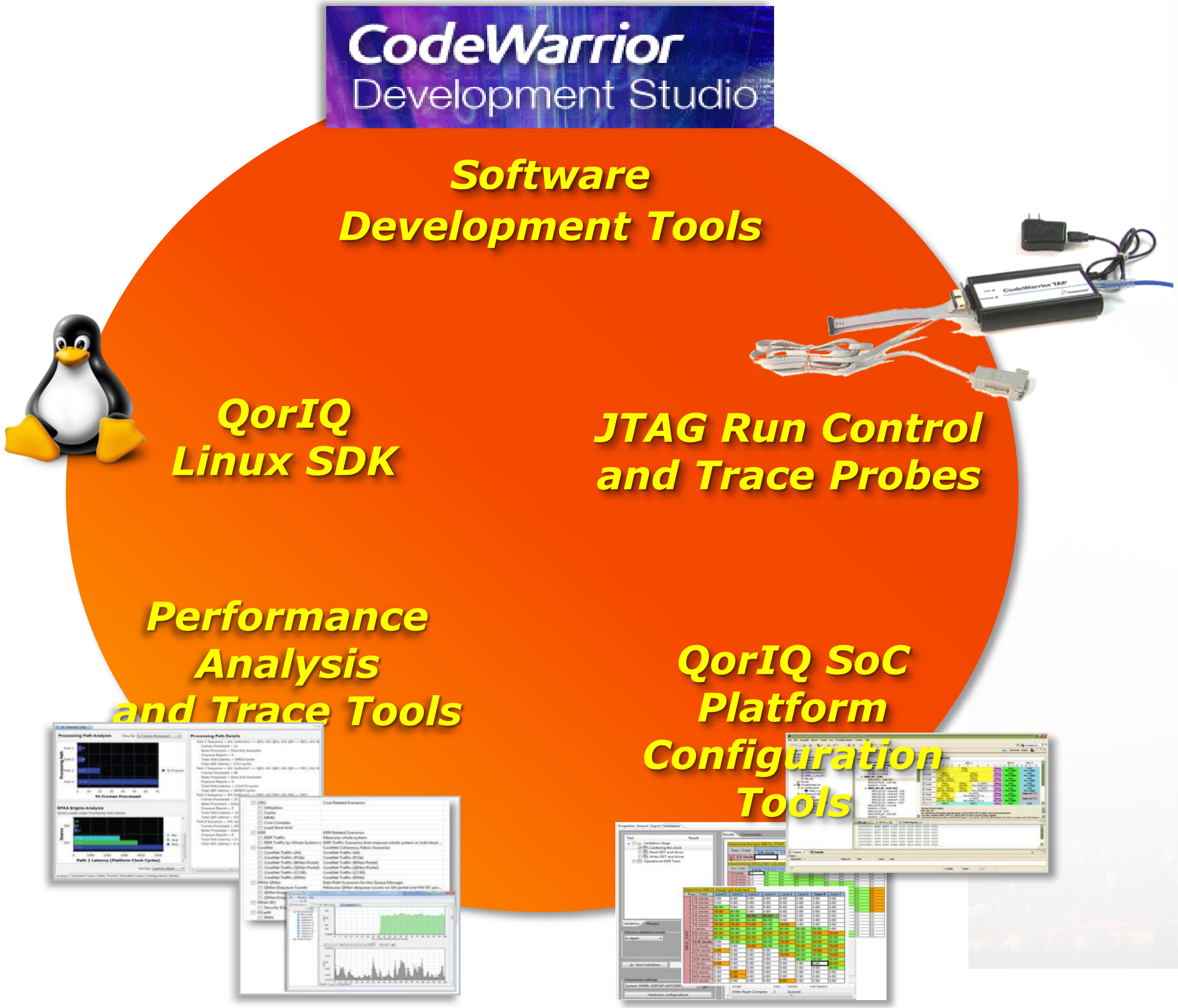
PUBLIC USE

AGENDA

- CodeWarrior Development Studio
- Introducing the LS2085A RDB
- Preparing the environment
- RDB-LS2085A with SDK EAR6.0
- Summary of CodeWarrior Software Analysis features
- Trace Compass
- Logging via DebugPrint
- U-boot tracing
- Linux user application tracing
- Linux trace – view results
- Smart filtering (tracepoints, ranges, modules)
- Summary

CODEWARRIOR DEVELOPMENT STUDIO

Layerscape LS2085A Software and Tools Enablement



CodeWarrior Development Studio

Software Development Tools

QorIQ Linux SDK

JTAG Run Control and Trace Probes

Performance Analysis and Trace Tools

QorIQ SoC Platform Configuration Tools

CodeWarrior Development Studio

A Complete Development Environment Under Eclipse

- **Eclipse IDE**
  - Configuration Wizards
  - Plug-In Architecture
  - 3rd party community

- **Build Tools**
  - C/C++ Compiler

- **Initialization Tools**
  - SOC platform initialization & configuration

- **Run Control**
  - CW-TAP

- **Debugger**
  - Multicore aware
  - Cross-triggering
    - Run/Stop of targets simultaneously
  - Access to all on-chip resources
  - Linux awareness

- **Software Analysis - Trace & Profile**
  - Leverages chip capabilities
    - Profiling Unit
    - In system trace buffering
  - Trace / Code / Performance Viewer
  - Offline trace visibility

CodeWarrior Aids Debug Through Multiple Phases

- Non-intrusive debug through trace
  - Core and SoC trace sources: configuration, extraction, visibility
  - Post-mortem debugging: offline trace
  - Debug-print
  - Linux aware trace
  - Linux application trace
  - Code Coverage

Software Products and Services

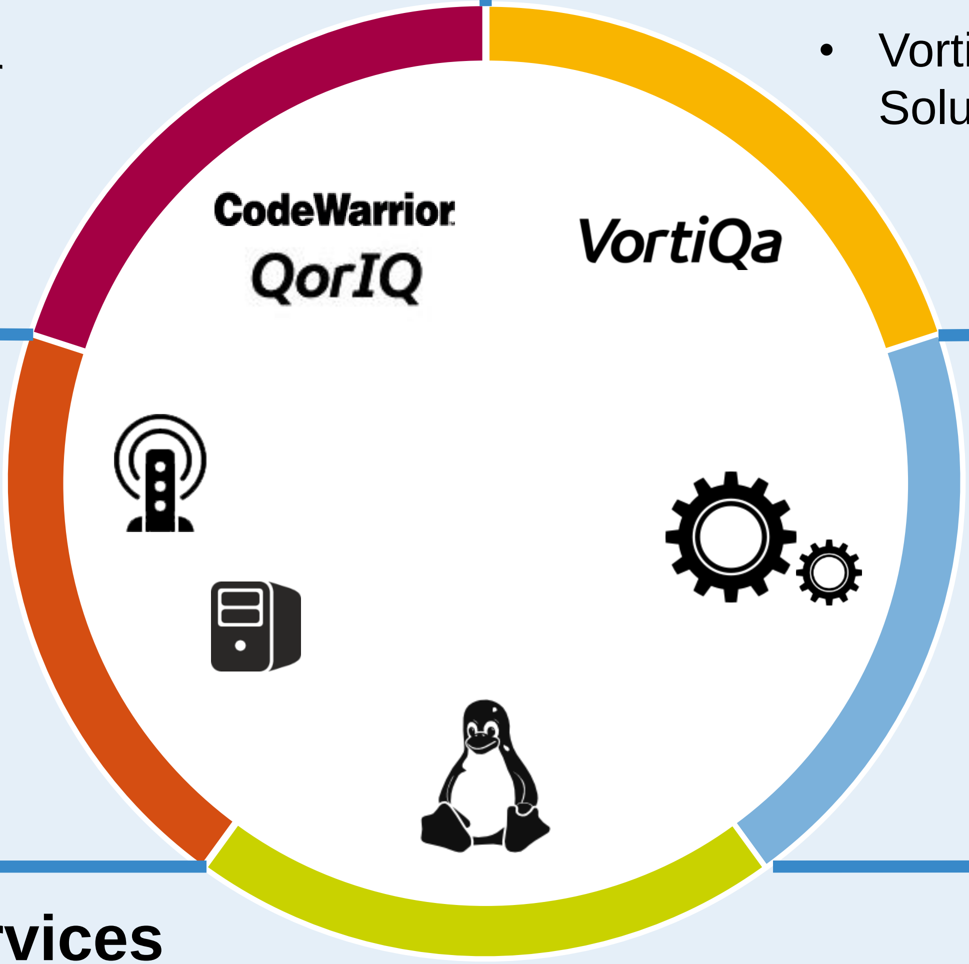
Visit us in the Tech Lab – #247

Development Tools

- CodeWarrior

Runtime Products

- VortiQa Software Solutions



Solutions Reference

- IOT Gateway
- OpenWRT+

Integration Services

- Security Consulting
- Hardened Linux

Linux® Services

- Commercial Support
- Performance Tuning

***Accelerate* Customer Time-to-Market**

***Deliver* Commercial Software, Support, Services and Solutions**

***Simplify* Software Engagement with NXP**

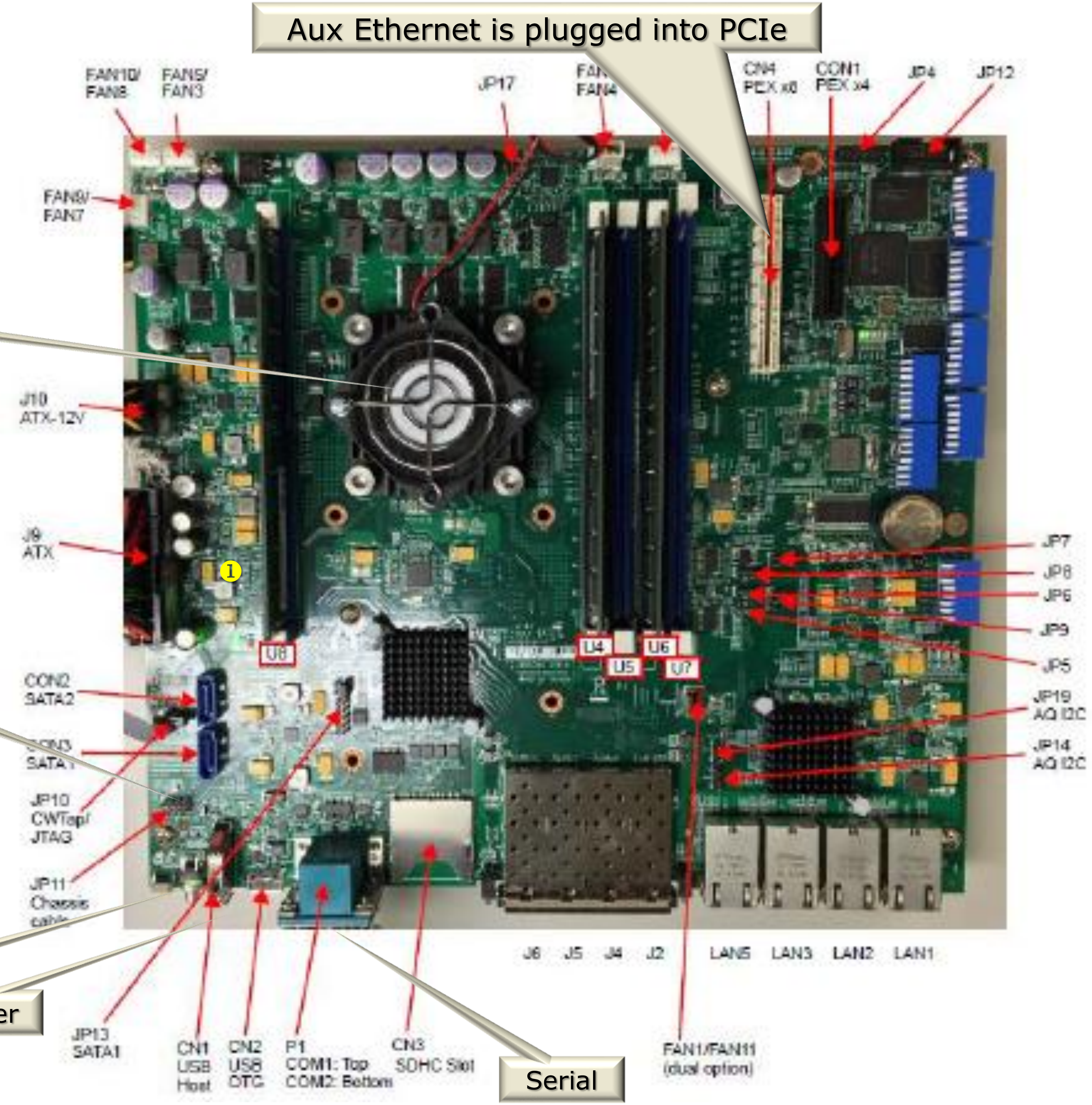
***Create* Success!**

Find us online at www.nxp.com/networking-services

INTRODUCING THE LS2085A RDB

LS2085A RDB

Top View



Aux Ethernet is plugged into PCIe

LS2085A

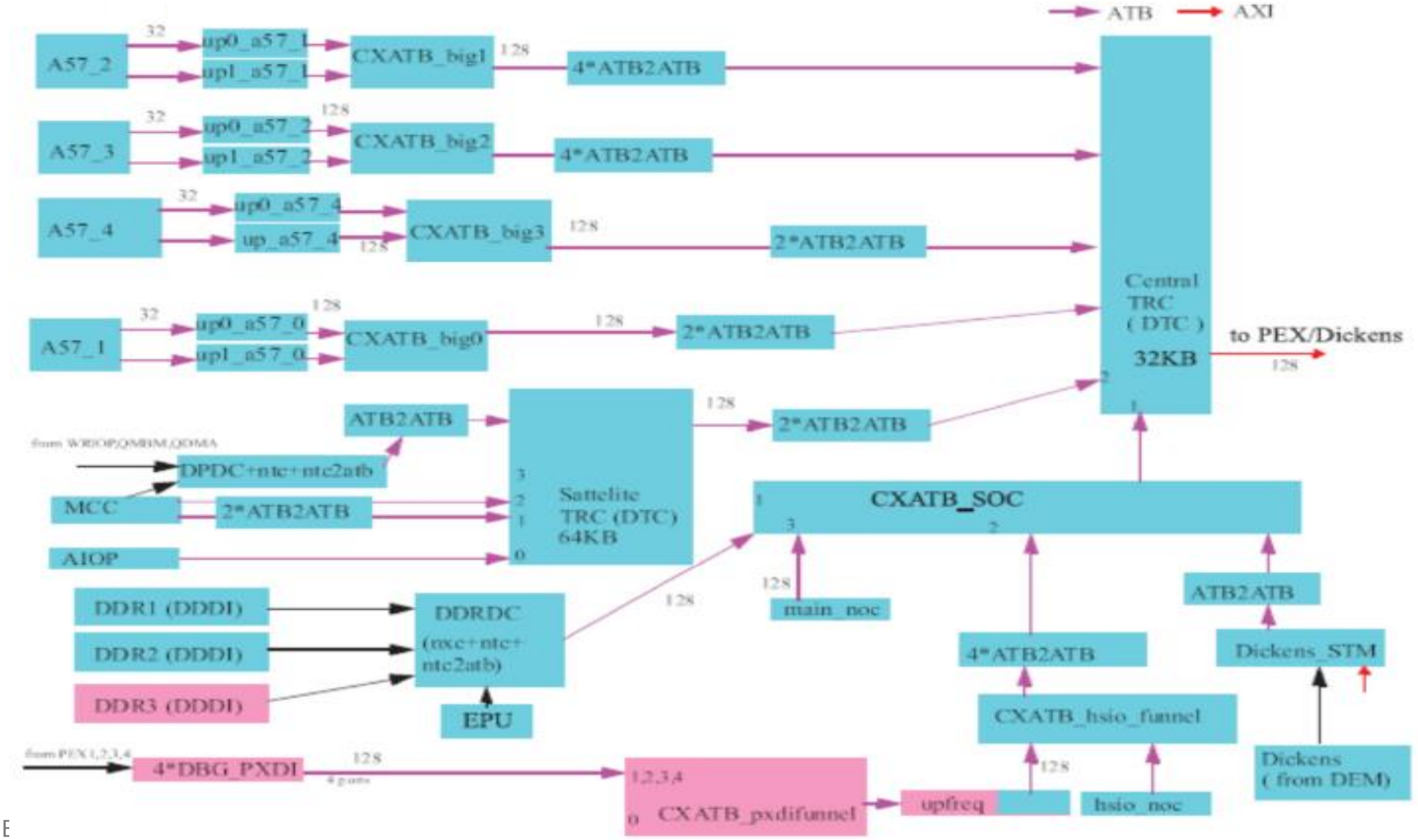
JTAG

RESET

Power

Serial

QorIQ LS2085A TraceIP Block Diagram



Debug Features

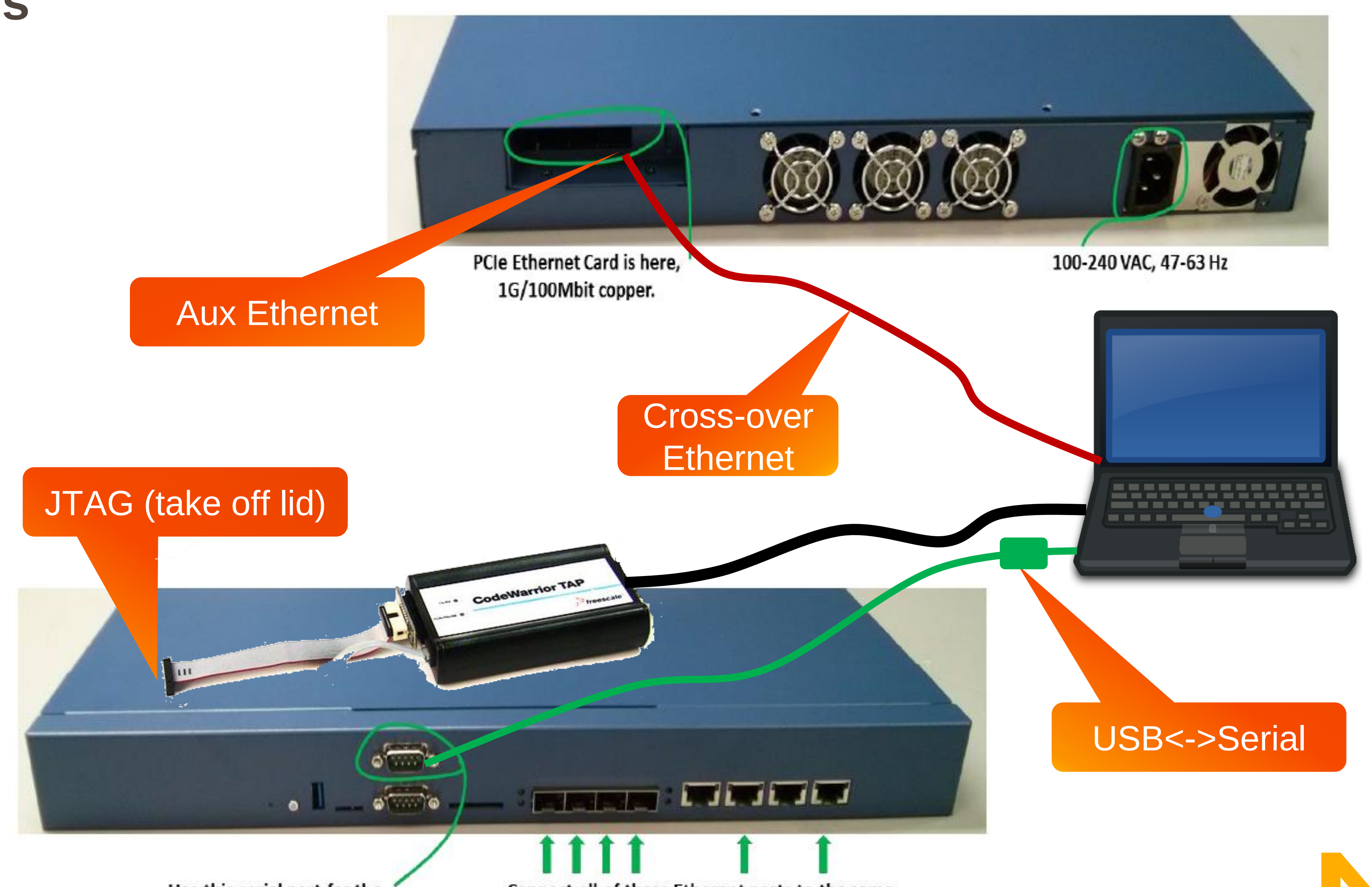
- Run-Control debug features in cores
  - Cross-triggering between cores

- Trace
  - Program trace (ETM)
  - System trace (STM)
  - Stored in internal memory or DDR
    - No external export via TPIU or Aurora

- EPU Performance Monitor

PREPARING THE ENVIRONMENT

Connections



PCIe Ethernet Card is here, 1G/100Mbit copper.

100-240 VAC, 47-63 Hz

Aux Ethernet

Cross-over Ethernet

JTAG (take off lid)

USB<->Serial

Use this serial port for the console, 115200-8-1-N.

Connect all of these Ethernet ports to the same port on the other RDB. All are 10G only.

Items That Have Been Setup For You

- Host OS
  - Best to use Linux on the host when developing Linux on the target
  - Multiple Linux OS supported
  - 64-bit Linux required
  - Used Mint 17.1 for class
- CodeWarrior for Networked Applications v2016.01
  - CodeWarrior for Layerscape ARMv8 ISA
- QorIQ Linux SDK for LS2085A RDB
  - Installed from ISOs – could also obtain from GIT
    - Layerscape2-SDK-AARCH64-IMAGE-20150515-yocto
    - Layerscape2-SDK-SOURCE-20150515-yocto
  - Did not use CACHE
  - Added extensions for tracing support

Items That Have Been Setup For You

- Install on host
  - Yocto
  - Minicom / cutecom
    - 115200-8-N-1
  - Tftp server (not used in class)
  - telnet / putty (not used in class)

- Read RDB Quickstart Guide!
- Bitbake the SDK
- Install on target
  - Flash U-boot

Class Information

- Linux Login
  - User: class
  - Password: codewarrior

- SDK is installed in ~/SDK
  - Need to use full path in tool: /home/class/SDK

- On desktop
  - Launcher to Codewarrior – looks like rocket
  - shortcut to cutecom
  - Menu has link to terminal
    - Use for launch minicom

- No password on target Linux

RDB-LS2085A

SDK EAR6.0 Installed on LS2085A RDB

U-Boot Startup Messages

- Reset the RDB-LS2085A, interrupt the countdown
- Review the u-boot output in the console window:

```
U-Boot 2015.10LS2085A-SDK+g3242b20 (Mar 21 2016 - 13:23:23 +0200)

SoC:  LS2085E (0x87010010)
Clock Configuration:
       CPU0(A57):1800 MHz  CPU1(A57):1800 MHz  CPU2(A57):1800 MHz
       CPU3(A57):1800 MHz  CPU4(A57):1800 MHz  CPU5(A57):1800 MHz
       CPU6(A57):1800 MHz  CPU7(A57):1800 MHz
       Bus:      600  MHz  DDR:      1866.667 MT/s     DP-DDR:   1600 MT/s
Reset Configuration Word (RCW):
       00: 48303830 48480048 00000000 00000000
       10: 00000000 00200000 00200000 00000000
       20: 01012980 00002580 00000000 00000000
       30: 00000e0b 00000000 00000000 00000000
       40: 00000000 00000000 00000000 00000000
       50: 00000000 00000000 00000000 00000000
       60: 00000000 00000000 00027000 00000000
       70: 412a0000 00000000 00000000 00000000
Model: Freescale Layerscape 2085a RDB Board
Board: LS2085E-RDB, Board Arch: V1, Board version: D, boot from vBank: 4
```

U-Boot Startup Messages

```
DDR    15 GiB (DDR4, 64-bit, CL=13, ECC on)
       DDR Controller Interleaving Mode: 256B
       DDR Chip-Select Interleaving Mode: CS0+CS1
DP-DDR 4 GiB (DDR4, 32-bit, CL=11, ECC on)
       DDR Chip-Select Interleaving Mode: CS0+CS1
Waking secondary cores to start from fff0b000
All (8) cores are up.
Using SERDES1 Protocol: 42 (0x2a)
Using SERDES2 Protocol: 65 (0x41)
Flash: 128 MiB
NAND:  2048 MiB
MMC:   FSL_SDHC: 0
AHCI 0001.0301 32 slots 1 ports 6 Gbps 0x1 impl SATA mode
flags: 64bit ncq pm clo only pmp fbss pio slum part ccc apst
Found 0 device(s).
SCSI:  Net:   crc32+
fsl-mc: Booting Management Complex ... SUCCESS
fsl-mc: Management Complex booted (version: 9.0.4, boot status: 0x1)
e1000: 68:05:ca:36:9c:7c
       DPMAC1@xgmii, DPMAC2@xgmii, DPMAC3@xgmii, DPMAC4@xgmii, DPMAC5@xgmii,
DPMAC6@xgmii, DPMAC7@xgmii, DPMAC8@xgmii, e1000#0 [PRIME]

Hit any key to stop autoboot:  0
```

Linux

- Linux is automatically booting
- If u-boot countdown has been interrupted, boot Linux with command “boot”
- When Linux booting is complete:
  - Login with user root and no password
  - Configure eth0 to 192.168.1.100

```
INIT: Entering runlevel: 5un-postinsts exists during rc.d purge
Configuring network interfaces... done.
Starting OpenBSD Secure Shell server: sshd
  generating ssh RSA key...
  generating ssh ECDSA key...
  generating ssh DSA key...
Poky (Yocto Project Reference Distro) 1.8.1 ls2085ardb /dev/ttyS1

ls2085ardb login: root
root@ls2085ardb:~# ifconfig eth0 192.168.1.100
root@ls2085ardb:~#
```

SUMMARY OF CW SOFTWARE ANALYSIS FEATURES

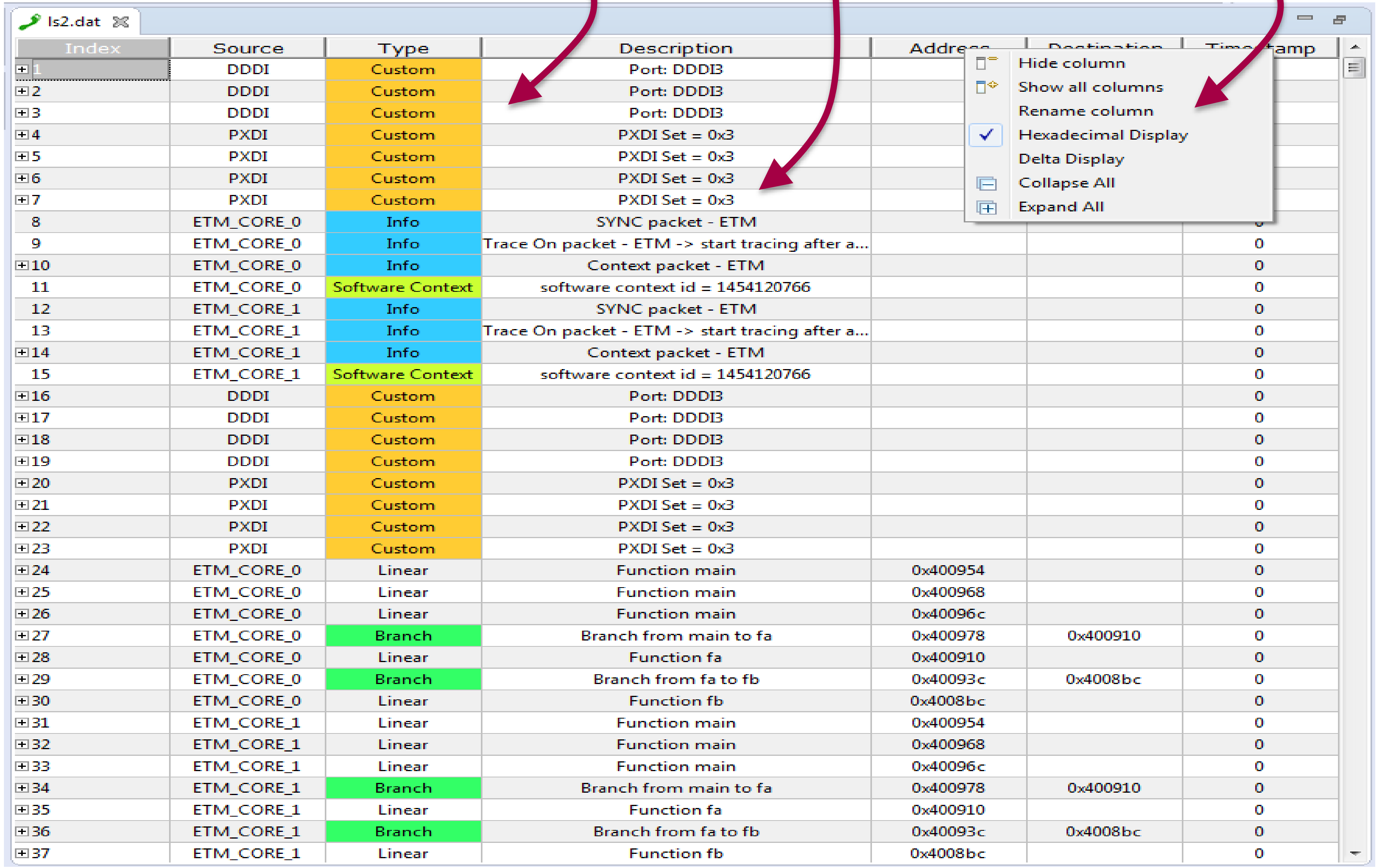
Trace overview

- Based on hardware trace modules that monitor and probe core execution, system busses, transactions, memory accesses, peripherals activity, etc.
- Minimal to no intrusiveness to system activity and performance
- Used to investigate crash analysis
- Assembly level instruction granularity for program trace
- Multiple collection modes supported (One Buffer, Overwrite)
- Multiple storage location for trace (Internal Buffer, DDR, Scatter-Gather, external device)
- Ability to filter trace events directly on target, multiple combinations
- Ability to combine multiple trace sources into one single stream
- Integrated at system level with hardware triggering mechanisms

Trace viewer screenshot

Various trace events from different sources

Customize the view



Index	Source	Type	Description	Address	Destination	Timestamp
1	DDDI	Custom	Port: DDDI3			
2	DDDI	Custom	Port: DDDI3			
3	DDDI	Custom	Port: DDDI3			
4	PXDI	Custom	PXDI Set = 0x3			
5	PXDI	Custom	PXDI Set = 0x3			
6	PXDI	Custom	PXDI Set = 0x3			
7	PXDI	Custom	PXDI Set = 0x3			
8	ETM_CORE_0	Info	SYNC packet - ETM			0
9	ETM_CORE_0	Info	Trace On packet - ETM -> start tracing after a...			0
10	ETM_CORE_0	Info	Context packet - ETM			0
11	ETM_CORE_0	Software Context	software context id = 1454120766			0
12	ETM_CORE_1	Info	SYNC packet - ETM			0
13	ETM_CORE_1	Info	Trace On packet - ETM -> start tracing after a...			0
14	ETM_CORE_1	Info	Context packet - ETM			0
15	ETM_CORE_1	Software Context	software context id = 1454120766			0
16	DDDI	Custom	Port: DDDI3			0
17	DDDI	Custom	Port: DDDI3			0
18	DDDI	Custom	Port: DDDI3			0
19	DDDI	Custom	Port: DDDI3			0
20	PXDI	Custom	PXDI Set = 0x3			0
21	PXDI	Custom	PXDI Set = 0x3			0
22	PXDI	Custom	PXDI Set = 0x3			0
23	PXDI	Custom	PXDI Set = 0x3			0
24	ETM_CORE_0	Linear	Function main	0x400954		0
25	ETM_CORE_0	Linear	Function main	0x400968		0
26	ETM_CORE_0	Linear	Function main	0x40096c		0
27	ETM_CORE_0	Branch	Branch from main to fa	0x400978	0x400910	0
28	ETM_CORE_0	Linear	Function fa	0x400910		0
29	ETM_CORE_0	Branch	Branch from fa to fb	0x40093c	0x4008bc	0
30	ETM_CORE_0	Linear	Function fb	0x4008bc		0
31	ETM_CORE_1	Linear	Function main	0x400954		0
32	ETM_CORE_1	Linear	Function main	0x400968		0
33	ETM_CORE_1	Linear	Function main	0x40096c		0
34	ETM_CORE_1	Branch	Branch from main to fa	0x400978	0x400910	0
35	ETM_CORE_1	Linear	Function fa	0x400910		0
36	ETM_CORE_1	Branch	Branch from fa to fb	0x40093c	0x4008bc	0
37	ETM_CORE_1	Linear	Function fb	0x4008bc		0

Hierarchical Profiler overview

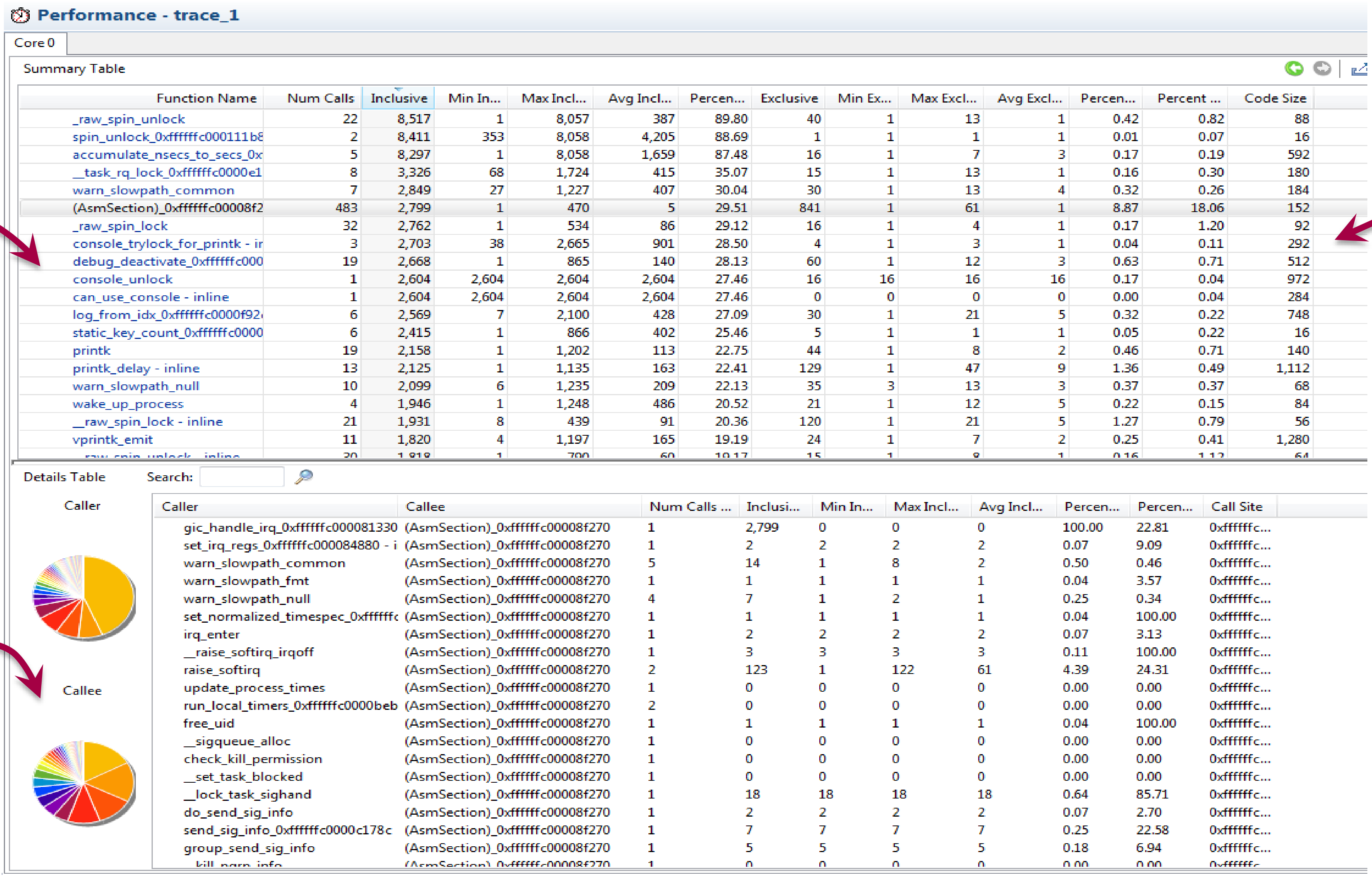
- Based on hardware trace
- Calculate inclusive(self) and exclusive(hierarchical) time for functions
- Min/Max/Average analysis
- Caller/callee breakdown (hierarchy of calls)
- Code optimization – according with Pareto principle “*80% of the effects come from 20% of the causes*”

Hierarchical Profiler viewer screenshot

Functions overview

Statistics

Caller-callee pairs



Function Name	Num Calls	Inclusive	Min In...	Max Incl...	Avg Incl...	Percen...	Exclusive	Min Ex...	Max Excl...	Avg Excl...	Percen...	Percent ...	Code Size
_raw_spin_unlock	22	8,517	1	8,057	387	89.80	40	1	13	1	0.42	0.82	88
spin_unlock_0xffffffc000111b8	2	8,411	353	8,058	4,205	88.69	1	1	1	1	0.01	0.07	16
accumulate_nsecs_to_secs_0x	5	8,297	1	8,058	1,659	87.48	16	1	7	3	0.17	0.19	592
__task_rq_lock_0xffffffc0000e1	8	3,326	68	1,724	415	35.07	15	1	13	1	0.16	0.30	180
warn_slowpath_common	7	2,849	27	1,227	407	30.04	30	1	13	4	0.32	0.26	184
(AsmSection)_0xffffffc00008f2	483	2,799	1	470	5	29.51	841	1	61	1	8.87	18.06	152
_raw_spin_lock	32	2,762	1	534	86	29.12	16	1	4	1	0.17	1.20	92
console_trylock_for_printk - ir	3	2,703	38	2,665	901	28.50	4	1	3	1	0.04	0.11	292
debug_deactivate_0xffffffc000	19	2,668	1	865	140	28.13	60	1	12	3	0.63	0.71	512
console_unlock	1	2,604	2,604	2,604	2,604	27.46	16	16	16	16	0.17	0.04	972
can_use_console - inline	1	2,604	2,604	2,604	2,604	27.46	0	0	0	0	0.00	0.04	284
log_from_idx_0xffffffc0000f92	6	2,569	7	2,100	428	27.09	30	1	21	5	0.32	0.22	748
static_key_count_0xffffffc0000	6	2,415	1	866	402	25.46	5	1	1	1	0.05	0.22	16
printk	19	2,158	1	1,202	113	22.75	44	1	8	2	0.46	0.71	140
printk_delay - inline	13	2,125	1	1,135	163	22.41	129	1	47	9	1.36	0.49	1,112
warn_slowpath_null	10	2,099	6	1,235	209	22.13	35	3	13	3	0.37	0.37	68
wake_up_process	4	1,946	1	1,248	486	20.52	21	1	12	5	0.22	0.15	84
__raw_spin_lock - inline	21	1,931	8	439	91	20.36	120	1	21	5	1.27	0.79	56
vprintk_emit	11	1,820	4	1,197	165	19.19	24	1	7	2	0.25	0.41	1,280
__raw_spin_unlock - inline	20	1,818	1	700	60	19.17	15	1	8	1	0.16	1.12	64

Caller	Callee	Num Calls ...	Inclusi...	Min In...	Max Incl...	Avg Incl...	Percen...	Percen...	Call Site
gic_handle_irq_0xffffffc000081330	(AsmSection)_0xffffffc00008f270	1	2,799	0	0	0	100.00	22.81	0xffffffc...
set_irq_regs_0xffffffc000084880 - i	(AsmSection)_0xffffffc00008f270	1	2	2	2	2	0.07	9.09	0xffffffc...
warn_slowpath_common	(AsmSection)_0xffffffc00008f270	5	14	1	8	2	0.50	0.46	0xffffffc...
warn_slowpath_fmt	(AsmSection)_0xffffffc00008f270	1	1	1	1	1	0.04	3.57	0xffffffc...
warn_slowpath_null	(AsmSection)_0xffffffc00008f270	4	7	1	2	1	0.25	0.34	0xffffffc...
set_normalized_timespec_0xffffffc	(AsmSection)_0xffffffc00008f270	1	1	1	1	1	0.04	100.00	0xffffffc...
irq_enter	(AsmSection)_0xffffffc00008f270	1	2	2	2	2	0.07	3.13	0xffffffc...
__raise_softirq_irqoff	(AsmSection)_0xffffffc00008f270	1	3	3	3	3	0.11	100.00	0xffffffc...
raise_softirq	(AsmSection)_0xffffffc00008f270	2	123	1	122	61	4.39	24.31	0xffffffc...
update_process_times	(AsmSection)_0xffffffc00008f270	1	0	0	0	0	0.00	0.00	0xffffffc...
run_local_timers_0xffffffc0000beb	(AsmSection)_0xffffffc00008f270	2	0	0	0	0	0.00	0.00	0xffffffc...
free_uid	(AsmSection)_0xffffffc00008f270	1	1	1	1	1	0.04	100.00	0xffffffc...
__sigqueue_alloc	(AsmSection)_0xffffffc00008f270	1	0	0	0	0	0.00	0.00	0xffffffc...
check_kill_permission	(AsmSection)_0xffffffc00008f270	1	0	0	0	0	0.00	0.00	0xffffffc...
__set_task_blocked	(AsmSection)_0xffffffc00008f270	1	0	0	0	0	0.00	0.00	0xffffffc...
__lock_task_sighand	(AsmSection)_0xffffffc00008f270	1	18	18	18	18	0.64	85.71	0xffffffc...
do_send_sig_info	(AsmSection)_0xffffffc00008f270	1	2	2	2	2	0.07	2.70	0xffffffc...
send_sig_info_0xffffffc0000c178c	(AsmSection)_0xffffffc00008f270	1	7	7	7	7	0.25	22.58	0xffffffc...
group_send_sig_info	(AsmSection)_0xffffffc00008f270	1	5	5	5	5	0.18	6.94	0xffffffc...
__kill_pgrp_info	(AsmSection)_0xffffffc00008f270	1	0	0	0	0	0.00	0.00	0xffffffc

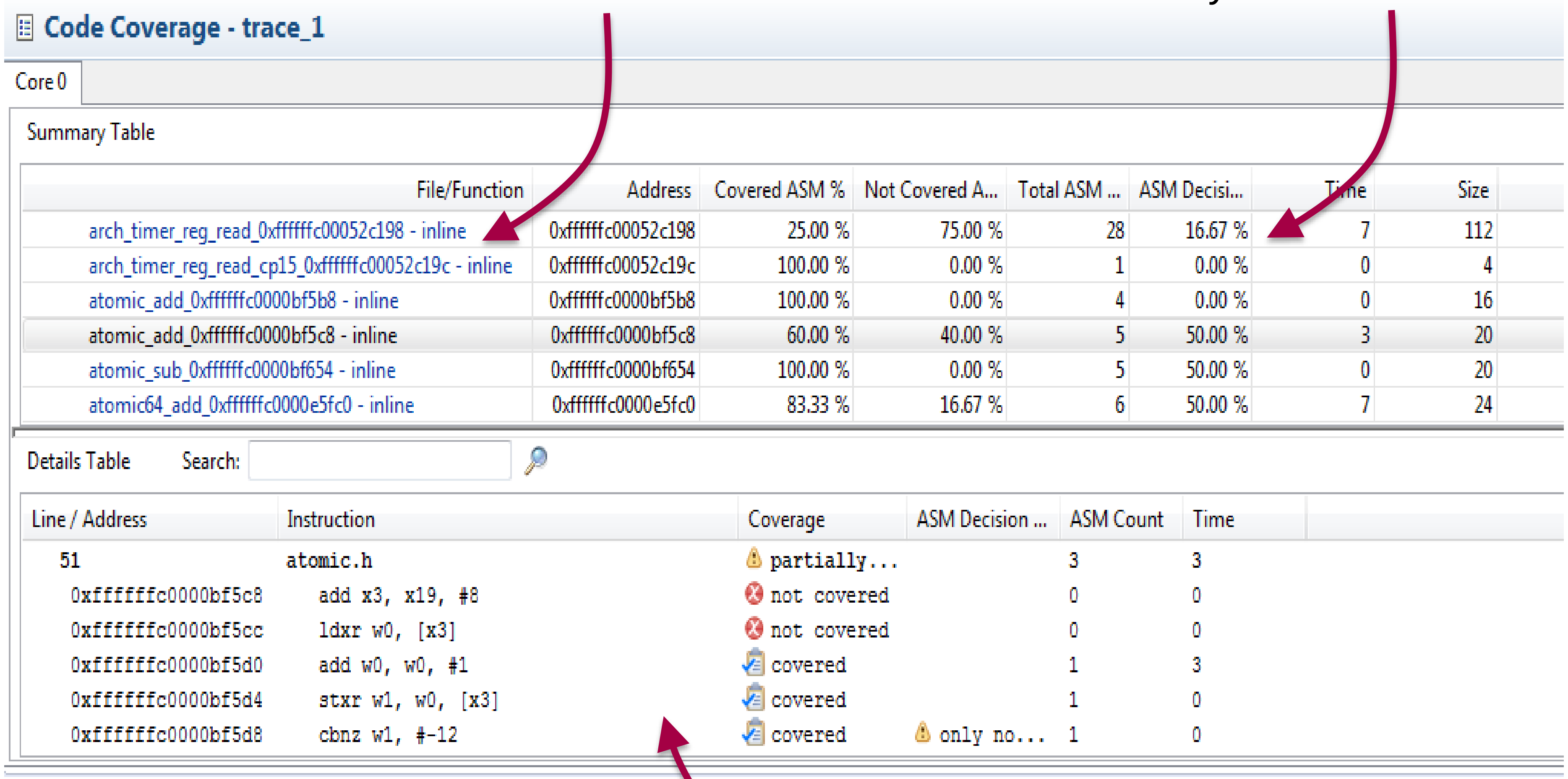
Code Coverage overview

- Based on hardware trace – no source instrumentation needed
- Provides coverage at assembly instruction level
- Statistics at assembly and C source line level
- Report in html format
- Decision coverage analysis at assembly instruction level

Code Coverage viewer screenshot

Summary table
for files and functions

Coverage metrics
at assembly and source level



Code Coverage - trace_1

Core 0

Summary Table

File/Function	Address	Covered ASM %	Not Covered A...	Total ASM ...	ASM Decisi...	Time	Size
arch_timer_reg_read_0xffffffc00052c198 - inline	0xffffffc00052c198	25.00 %	75.00 %	28	16.67 %	7	112
arch_timer_reg_read_cp15_0xffffffc00052c19c - inline	0xffffffc00052c19c	100.00 %	0.00 %	1	0.00 %	0	4
atomic_add_0xffffffc0000bf5b8 - inline	0xffffffc0000bf5b8	100.00 %	0.00 %	4	0.00 %	0	16
atomic_add_0xffffffc0000bf5c8 - inline	0xffffffc0000bf5c8	60.00 %	40.00 %	5	50.00 %	3	20
atomic_sub_0xffffffc0000bf654 - inline	0xffffffc0000bf654	100.00 %	0.00 %	5	50.00 %	0	20
atomic64_add_0xffffffc0000e5fc0 - inline	0xffffffc0000e5fc0	83.33 %	16.67 %	6	50.00 %	7	24

Details Table Search:

Line / Address	Instruction	Coverage	ASM Decision ...	ASM Count	Time
51	atomic.h	⚠ partially...		3	3
0xffffffc0000bf5c8	add x3, x19, #8	✖ not covered		0	0
0xffffffc0000bf5cc	ldxr w0, [x3]	✖ not covered		0	0
0xffffffc0000bf5d0	add w0, w0, #1	✓ covered		1	3
0xffffffc0000bf5d4	stxr w1, w0, [x3]	✓ covered		1	0
0xffffffc0000bf5d8	cbnz w1, #-12	✓ covered	⚠ only no...	1	0

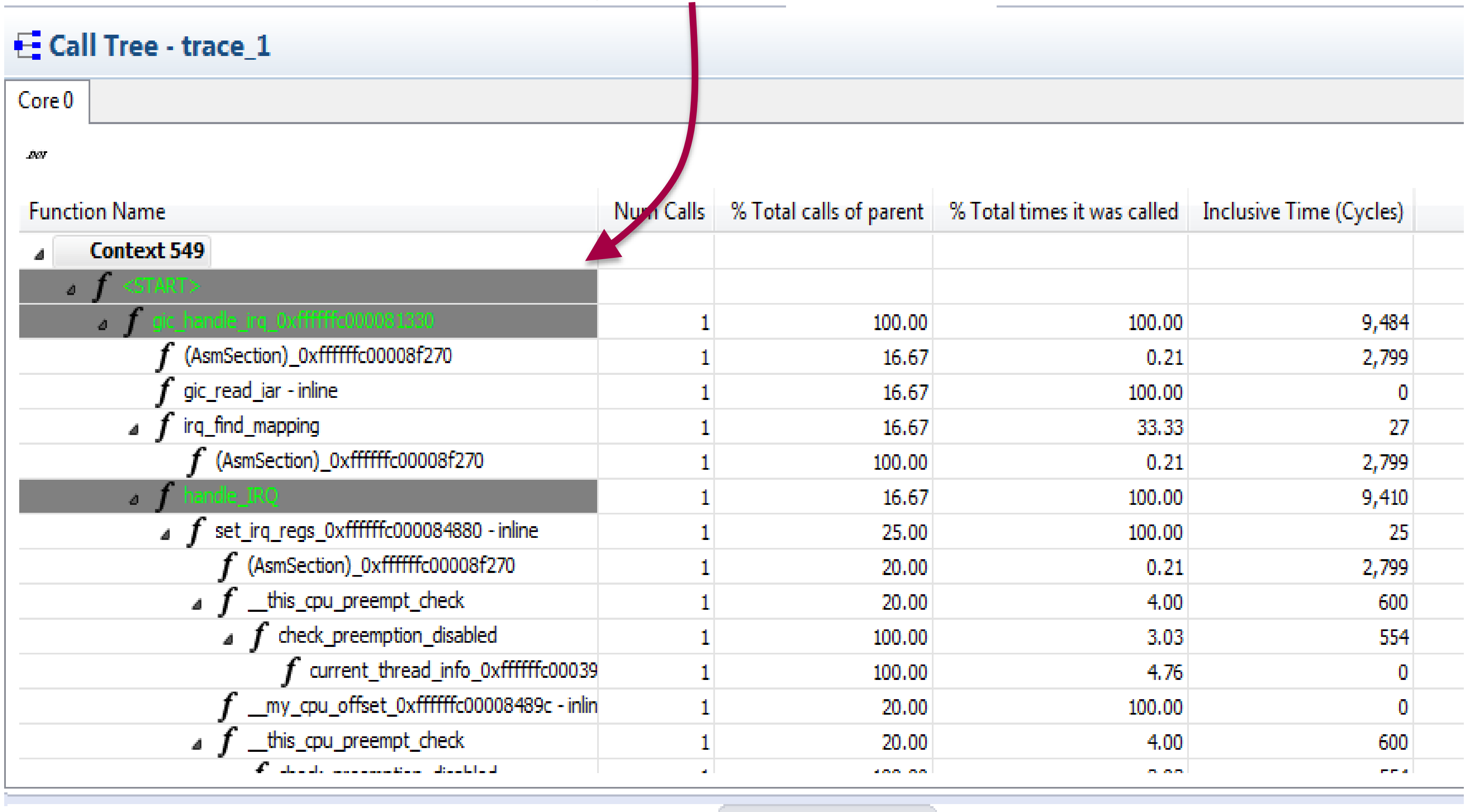
Coverage details
with asm decision coverage

Call Tree overview

- Based on hardware trace
- Identifies the longest calls path (critical path)
- Shows the max stack size (simulator)
- Investigate a certain flow

Call Tree viewer screenshot

Critical call chain
longest call stack



Call Tree - trace_1

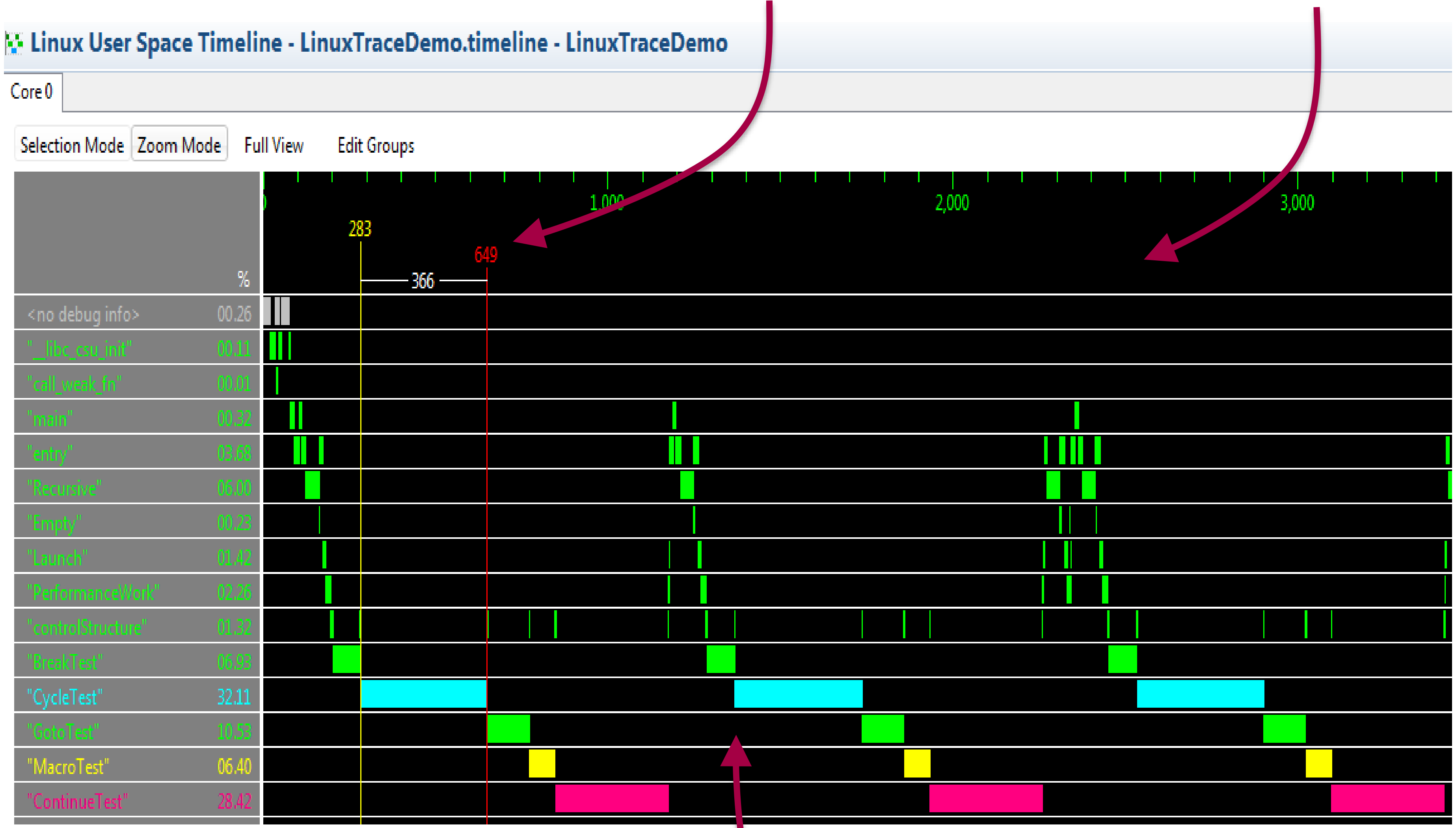
Core 0

Function Name	Num Calls	% Total calls of parent	% Total times it was called	Inclusive Time (Cycles)
Context 549				
<START>				
gic_handle_irq_0xffffffc000081330	1	100.00	100.00	9,484
(AsmSection)_0xffffffc00008f270	1	16.67	0.21	2,799
gic_read_iar - inline	1	16.67	100.00	0
irq_find_mapping	1	16.67	33.33	27
(AsmSection)_0xffffffc00008f270	1	100.00	0.21	2,799
handle_IRQ	1	16.67	100.00	9,410
set_irq_regs_0xffffffc000084880 - inline	1	25.00	100.00	25
(AsmSection)_0xffffffc00008f270	1	20.00	0.21	2,799
__this_cpu_preempt_check	1	20.00	4.00	600
check_preemption_disabled	1	100.00	3.03	554
current_thread_info_0xffffffc00039	1	100.00	4.76	0
__my_cpu_offset_0xffffffc00008489c - inlin	1	20.00	100.00	0
__this_cpu_preempt_check	1	20.00	4.00	600
check_preemption_disabled	1	100.00	3.03	554

Timeline overview

- Based on hardware trace
- Analyze execution flow
- Spot performance problems in code and bottlenecks
- Easily see out-of-order execution
- Understand the context of a certain execution error
- Logic analyzer look and feel
- Ability to group multiple functions to a single entry (e.g., module, unit)
- Customize the colors

Timeline viewer screenshot



	%
<no debug info>	00.26
"__libc_csu_init"	00.11
"call_weak_fn"	00.01
"main"	00.32
"entry"	03.68
"Recursive"	06.00
"Empty"	00.23
"Launch"	01.42
"PerformanceWork"	02.26
"controlStructure"	01.32
"BreakTest"	06.93
"CycleTest"	32.11
"GotoTest"	10.53
"MacroTest"	06.40
"ContinueTest"	28.42

Markers
for fast measurements

Execution flow
at glance

Change colors
for better visibility

TRACE COMPASS

Linux Tools – LTTng

- Linux Trace Toolkit – next generation: kernel and user-space tracer with view and analysis tools.

- LTTNG has been separated out of the Linux Trace Toolkit. Now a separate project called Trace Compass.
  - http://projects.eclipse.org/projects/tools.tracecompass

LTTNG

- Trace Compass is a Eclipse tool for viewing and analyzing any type of logs or traces.
  - Provide views, graphs, metrics, etc. to help extract useful information from traces, in a way that is more user-friendly and informative than huge text dumps

- Eclipse: “LTTng Kernel” perspective
- View the results
  - Events: timestamp, trace, Marker, Content
  - Histogram: trace event distribution in time
  - Control flow: processes list and their state in time
  - Resources: CPU resources per interrupts type
  - Statistics: event counters cpu time, cumulative /elapsed time
- Import or create a LTTng trace

Traces / Logs

- Trace Compass supports many trace formats:
  - Common Trace Format (CTF), including but not limited to:
    - Linux LTTng kernel traces
    - Linux LTTng-UST userspace traces
    - Linux Perf traces (using the out-of-tree patchset to convert to CTF)
  - GDB traces for debugging
  - The libpcap (PAcket CAPture) format, for network traces

Linux Trace

- Static probe points strategically located inside the kernel code
- Register/unregister with tracepoints via callback mechanism
- Can be used to profile, debug and understand kernel behavior

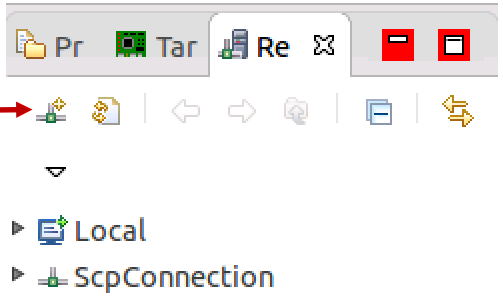
- Trace synchronization
  - Time correction
  - Multi-core
  - Dependency analysis, delay analyzer
  - Dependencies among processes

ACTIVITY

Trace Compass – new RSE connection

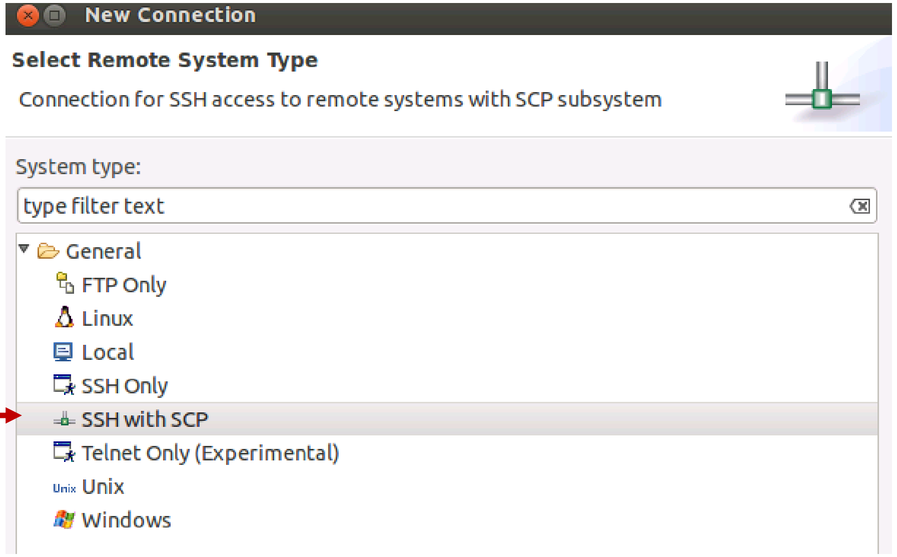
1. Open Remote Systems view (Window->Show View->Other->Remote Systems->Remote Systems)

New RSE connection



2. Create a Linux based RSE connection

SSH with SCP



New Connection

Select Remote System Type

Connection for SSH access to remote systems with SCP subsystem

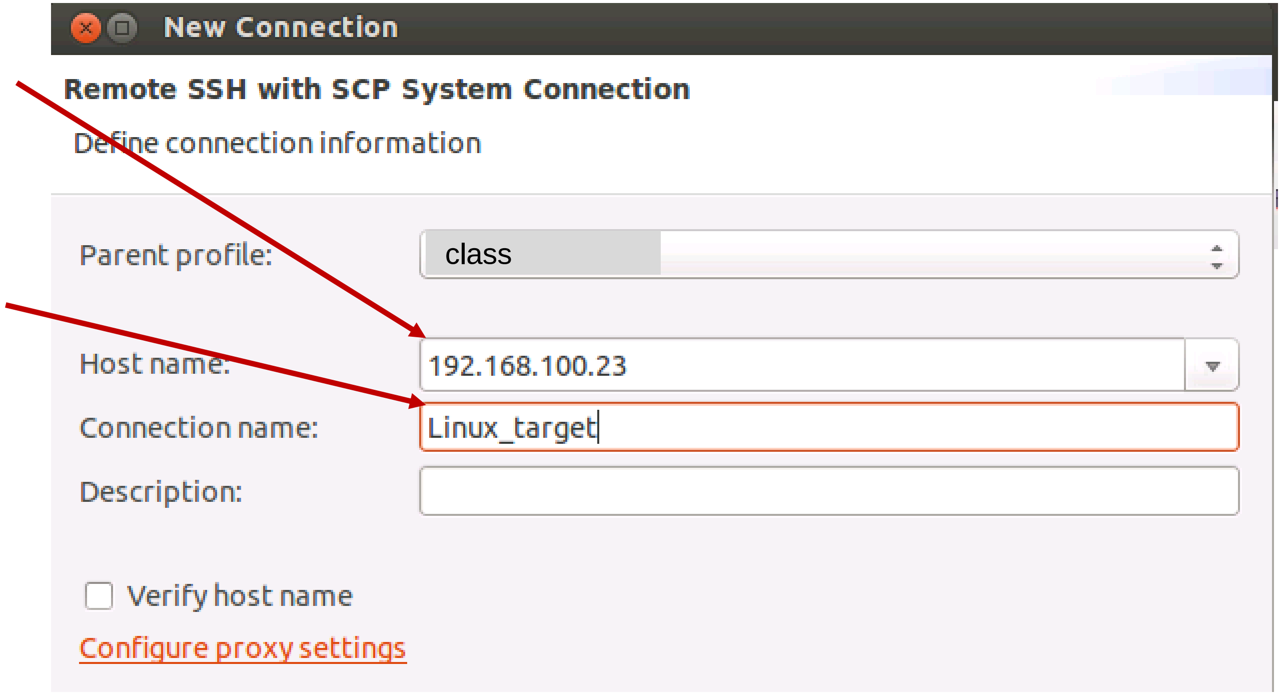
System type:

type filter text

- General
  - FTP Only
  - Linux
  - Local
  - SSH Only
  - SSH with SCP
  - Telnet Only (Experimental)
  - Unix
  - Windows

Trace Compass – new RSE connection

3. Follow the steps to create the RSE connection over SSH



Remote SSH with SCP System Connection

Define connection information

Parent profile: class

Host name: 192.168.100.23

Connection name: Linux_target

Description:

□ Verify host name

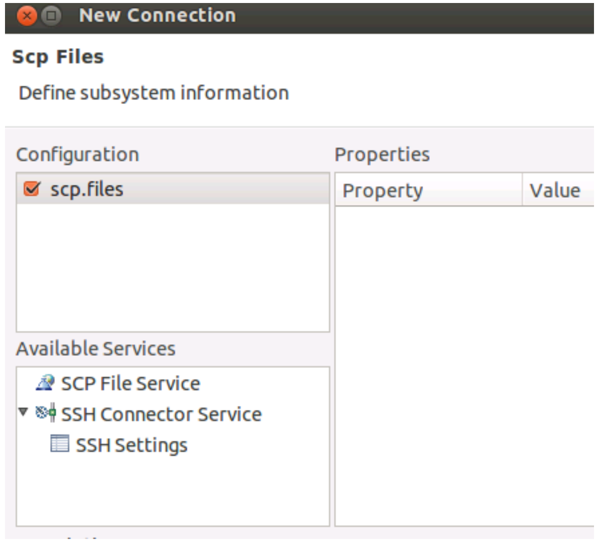
Configure proxy settings

IP of target

Name the connection

Trace Compass – new RSE connection

4. Continue to follow RSE connection creation wizard



New Connection

Scp Files

Define subsystem information

Configuration

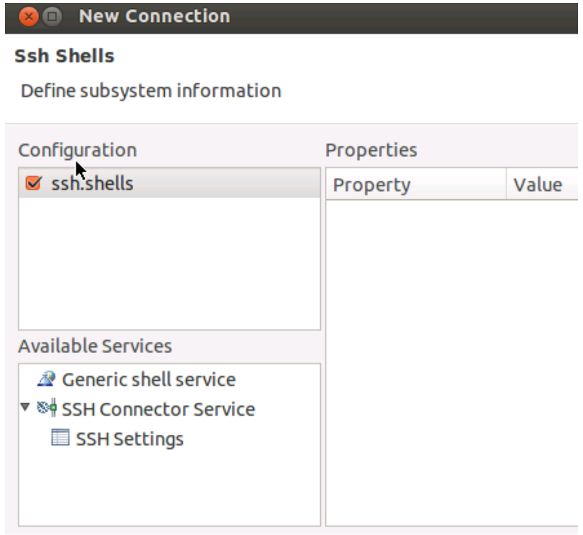
☑ scp.files

Properties

Property	Value

Available Services

- SCP File Service
- ▾ SSH Connector Service
  - SSH Settings



New Connection

Ssh Shells

Define subsystem information

Configuration

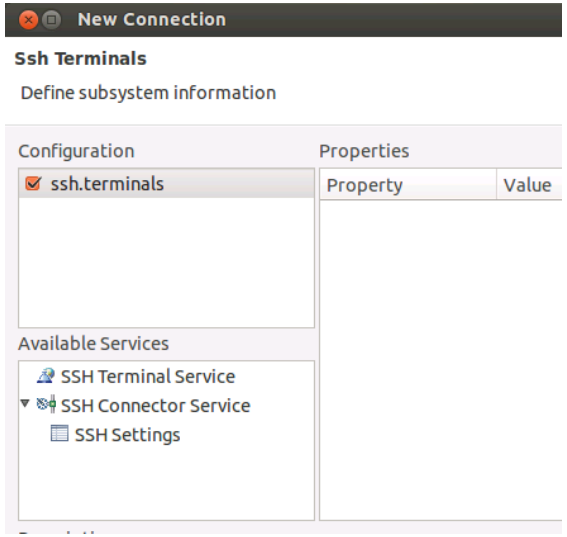
☑ ssh.shells

Properties

Property	Value

Available Services

- Generic shell service
- ▾ SSH Connector Service
  - SSH Settings



New Connection

Ssh Terminals

Define subsystem information

Configuration

☑ ssh.terminals

Properties

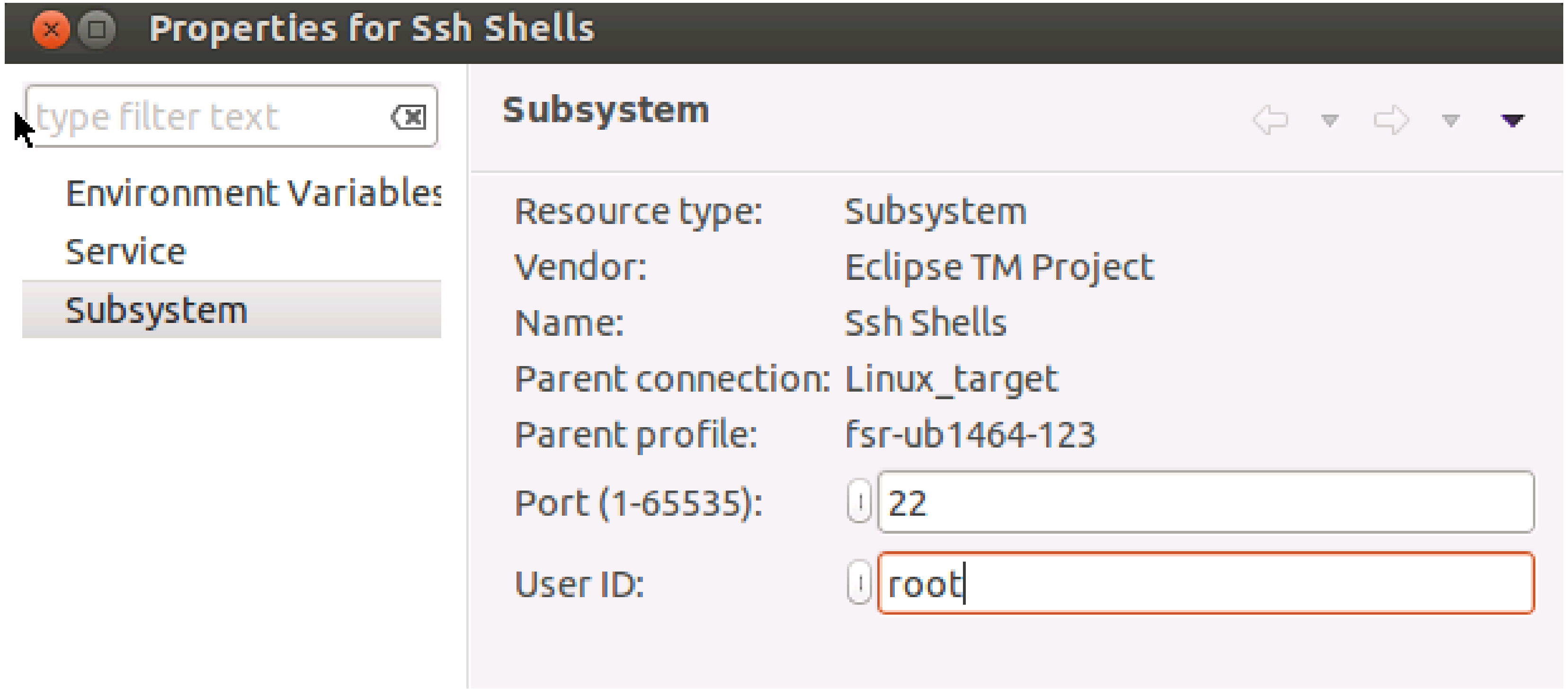
Property	Value

Available Services

- SSH Terminal Service
- ▾ SSH Connector Service
  - SSH Settings

Trace Compass – new RSE connection

5. Right-click on Ssh Shells -> Properties -> Subsystem. Verify the port (default is 22; change if port is forward). Set *root* as user ID.



Properties for Ssh Shells

Subsystem

Resource type: Subsystem

Vendor: Eclipse TM Project

Name: Ssh Shells

Parent connection: Linux_target

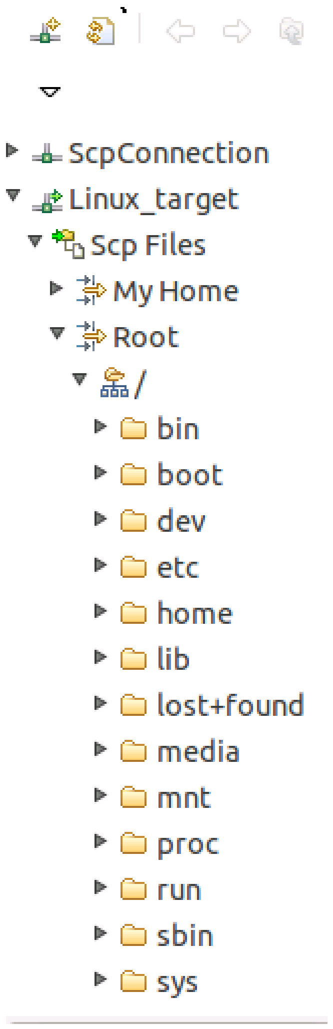
Parent profile: fsr-ub1464-123

Port (1-65535): 22

User ID: root

Trace Compass – new RSE connection

6. Now expand the Scp Files node and you will be able to browse the target file system:



```
▸ ScpConnection
▾ Linux_target
  ▾ Scp Files
    ▸ My Home
    ▾ Root
      ▾ /
        ▸ bin
        ▸ boot
        ▸ dev
        ▸ etc
        ▸ home
        ▸ lib
        ▸ lost+found
        ▸ media
        ▸ mnt
        ▸ proc
        ▸ run
        ▸ sbin
        ▸ sys
```

Trace Compass – trace session

1. Open a Terminal over a RSE connection from CodeWarrior (right-click on RSE tree and LaunchTerminal)
2. Load LTTng modules:

   modprobe lttng-tracer
3. Check that LTTng modules are loaded:

   lsmod
4. Create a new LTTng session:

   lttng create ftfSession
5. Enable all events for Kernel tracing:

   lttng enable-event --kernel --all
6. Start tracing session:

   lttng start
7. Run some applications (e.g., **ls**, **top**)
8. Stop tracing session:

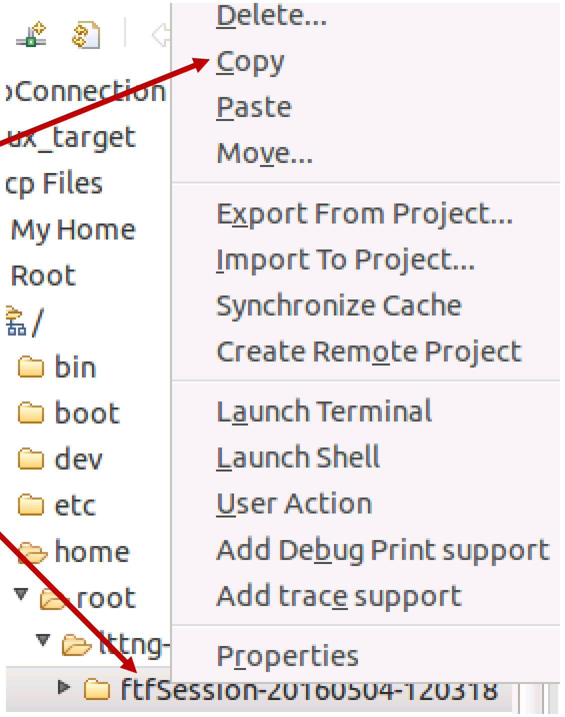
   lttng stop
9. Destroy session:

   lttng destroy

Trace Compass – trace session

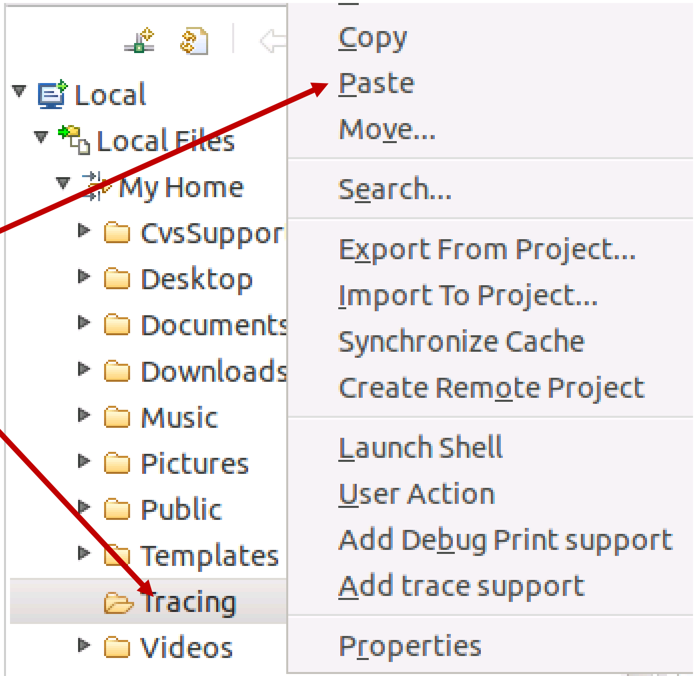
10. Notice the newly created folder in your home dir (*lttng-traces*)

11. Copy the session folder like this:



Copy … from this directory

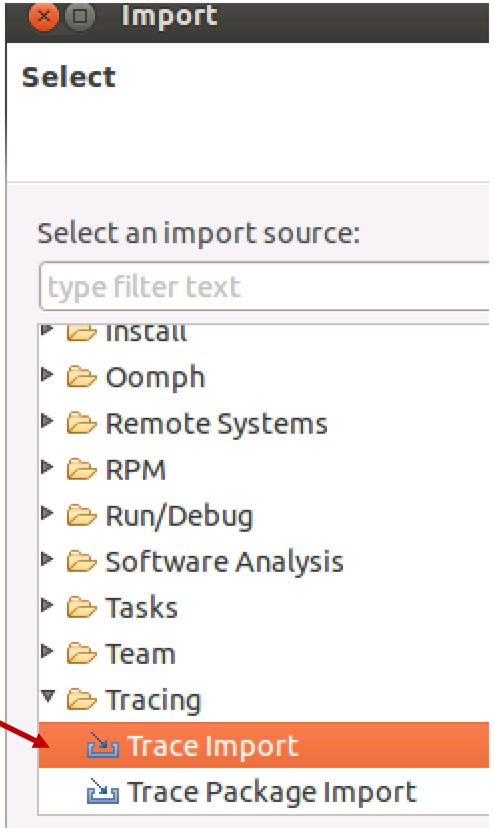
12. Paste in Local node (RSE):



Paste … to this directory

Trace Compass – trace session

13. Open *Project Explorer* view
14. Right-click and choose *Import*
15. Select *Tracing->Trace Import*



Import

Select

Select an import source:

type filter text

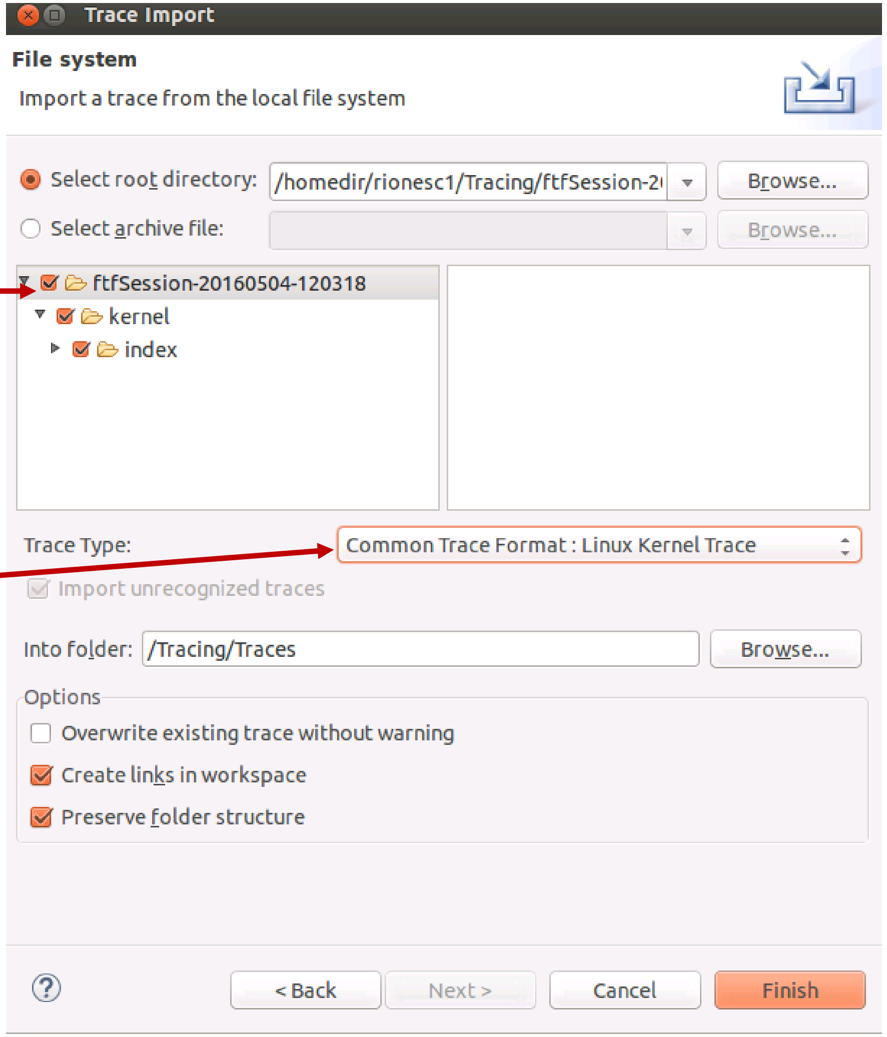
Import trace

Trace Compass – trace session

16. Choose the copied folder with trace session; check the file to import; select *Trace Type* as *LTTng Kernel Trace*

Select session file

Select Trace Type



Trace Import

File system

Import a trace from the local file system

● Select root directory: /homedir/rionesc1/Tracing/ftfSession-2([Browse...]

○ Select archive file: [Browse...]

- ☑ ftfSession-20160504-120318
  - ☑ kernel
    - ☑ index

Trace Type: Common Trace Format : Linux Kernel Trace

☑ Import unrecognized traces

Into folder: /Tracing/Traces [Browse...]

Options

☐ Overwrite existing trace without warning

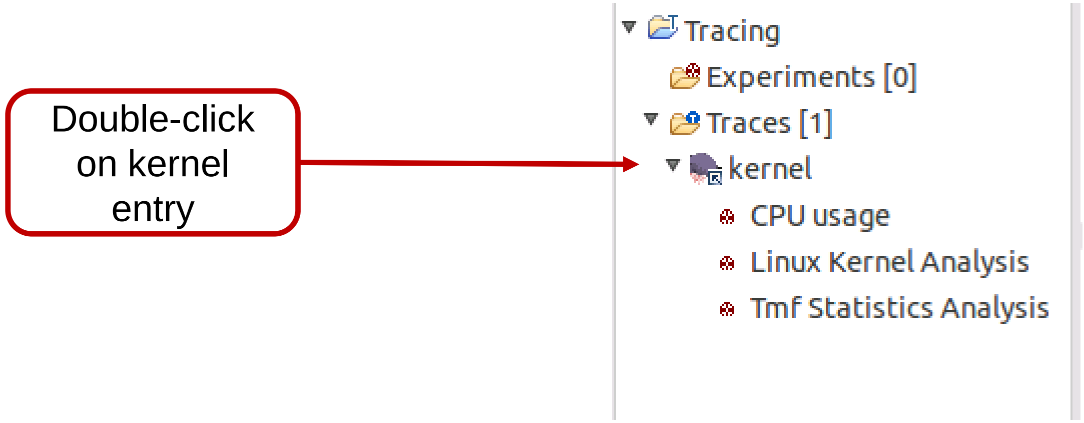
☑ Create links in workspace

☑ Preserve folder structure

Trace Compass – trace session

17. Open *LTTng Kernel* perspective

18. Double-click on imported trace session from *Project Explorer* view (kernel entry):



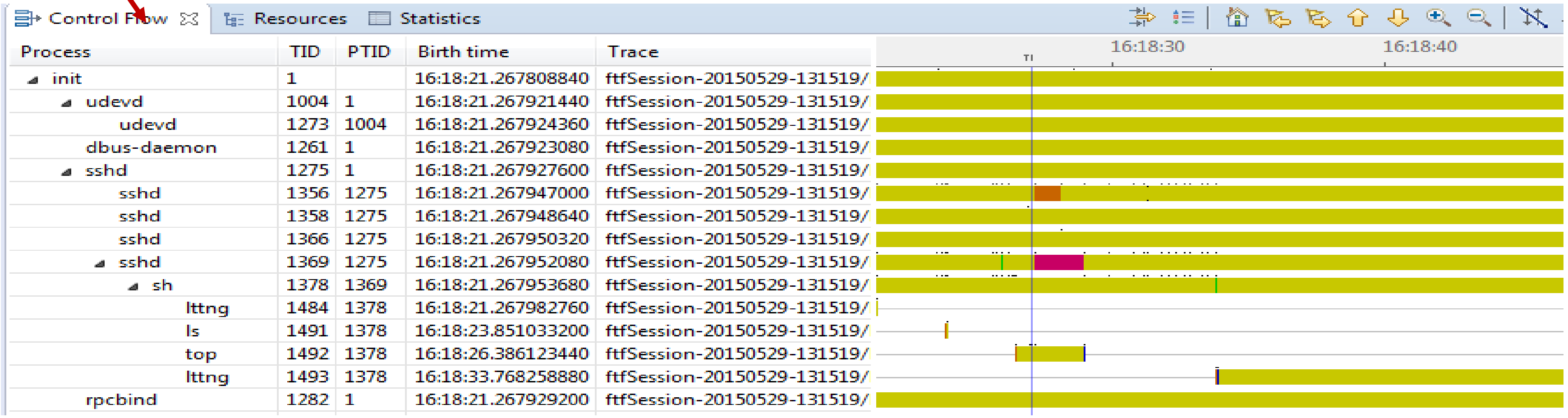
Double-click on kernel entry

- Tracing
  - Experiments [0]
  - Traces [1]
    - kernel
      - CPU usage
      - Linux Kernel Analysis
      - Tmf Statistics Analysis

19. Various views will open and you can explore trace results

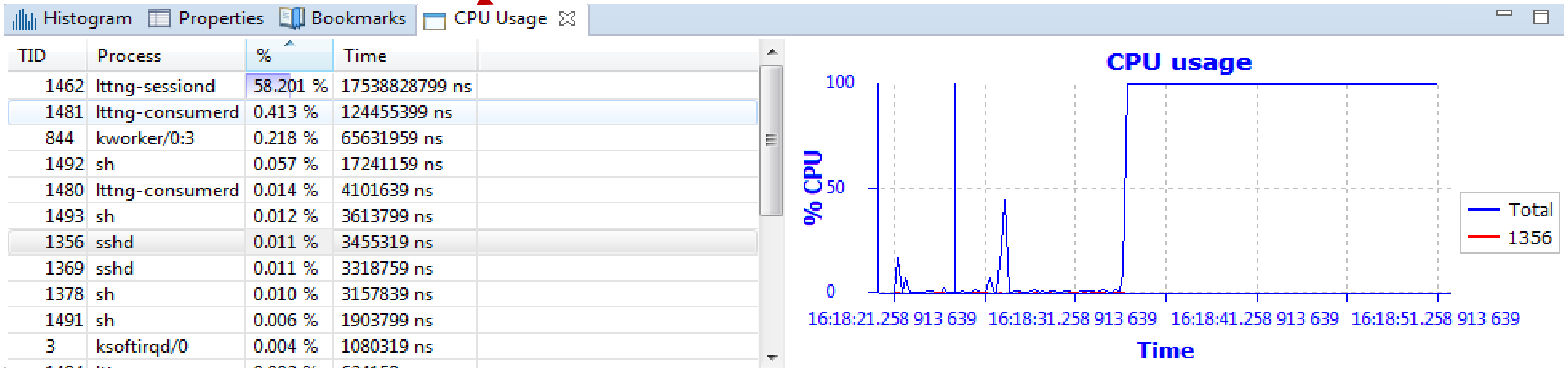
Trace Compass views

Control Flow



Process	TID	PTID	Birth time	Trace
init	1		16:18:21.267808840	ftfSession-20150529-131519/
udevd	1004	1	16:18:21.267921440	ftfSession-20150529-131519/
udevd	1273	1004	16:18:21.267924360	ftfSession-20150529-131519/
dbus-daemon	1261	1	16:18:21.267923080	ftfSession-20150529-131519/
sshd	1275	1	16:18:21.267927600	ftfSession-20150529-131519/
sshd	1356	1275	16:18:21.267947000	ftfSession-20150529-131519/
sshd	1358	1275	16:18:21.267948640	ftfSession-20150529-131519/
sshd	1366	1275	16:18:21.267950320	ftfSession-20150529-131519/
sshd	1369	1275	16:18:21.267952080	ftfSession-20150529-131519/
sh	1378	1369	16:18:21.267953680	ftfSession-20150529-131519/
lttng	1484	1378	16:18:21.267982760	ftfSession-20150529-131519/
ls	1491	1378	16:18:23.851033200	ftfSession-20150529-131519/
top	1492	1378	16:18:26.386123440	ftfSession-20150529-131519/
lttng	1493	1378	16:18:33.768258880	ftfSession-20150529-131519/
rpcbind	1282	1	16:18:21.267929200	ftfSession-20150529-131519/

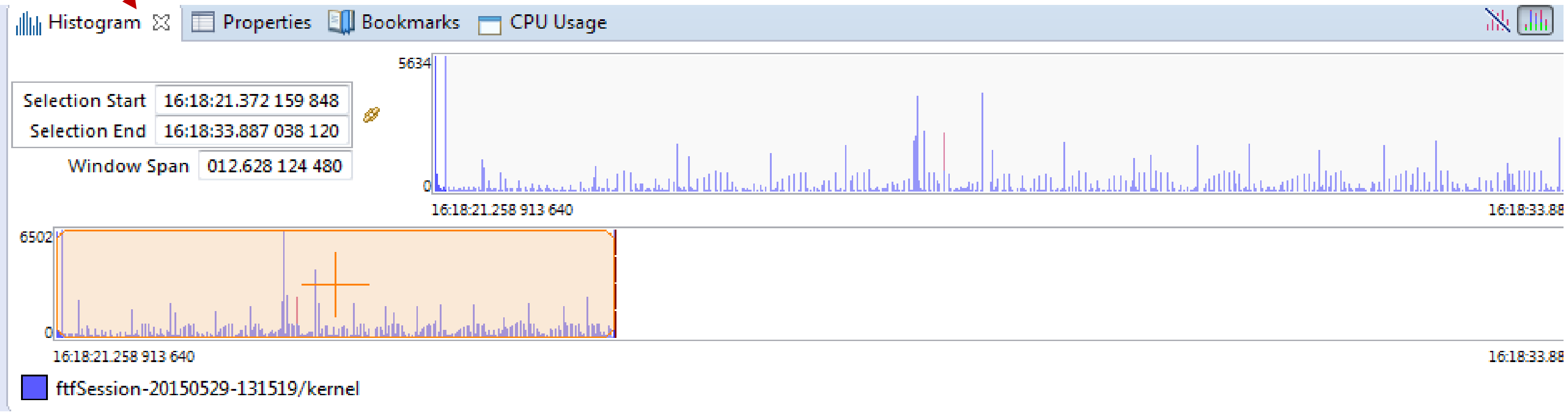
CPU Usage



TID	Process	%	Time
1462	lttng-sessiond	58.201 %	17538828799 ns
1481	lttng-consumerd	0.413 %	124455399 ns
844	kworker/0:3	0.218 %	65631959 ns
1492	sh	0.057 %	17241159 ns
1480	lttng-consumerd	0.014 %	4101639 ns
1493	sh	0.012 %	3613799 ns
1356	sshd	0.011 %	3455319 ns
1369	sshd	0.011 %	3318759 ns
1378	sh	0.010 %	3157839 ns
1491	sh	0.006 %	1903799 ns
3	ksoftirqd/0	0.004 %	1080319 ns

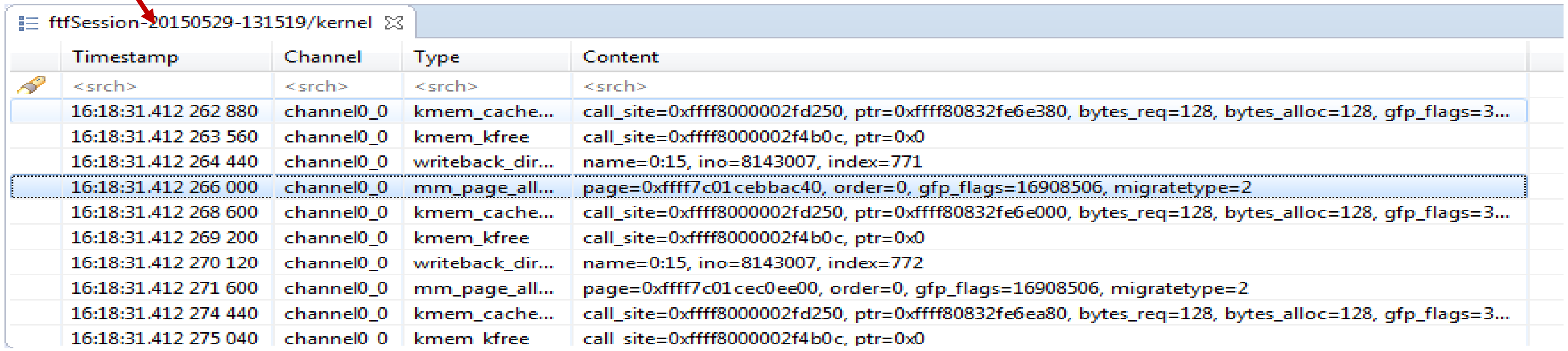
Trace Compass views

Histogram



Selection Start 16:18:21.372 159 848
Selection End 16:18:33.887 038 120
Window Span 012.628 124 480

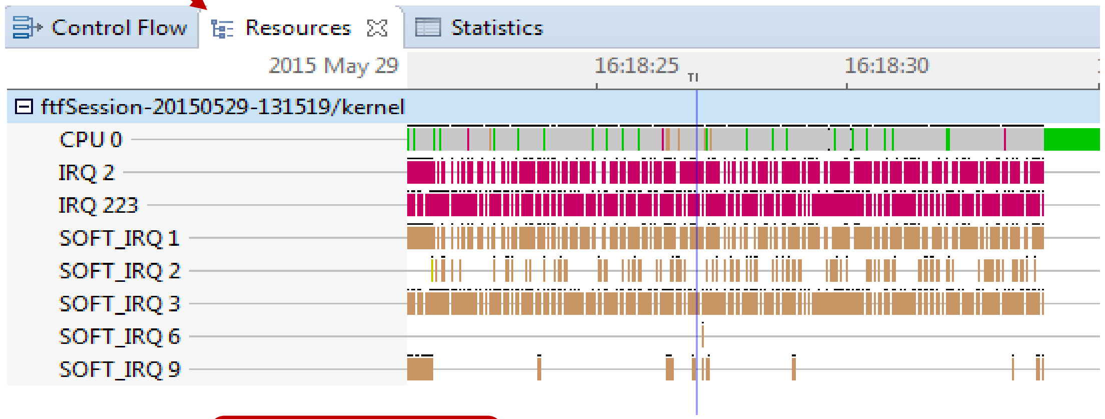
Events



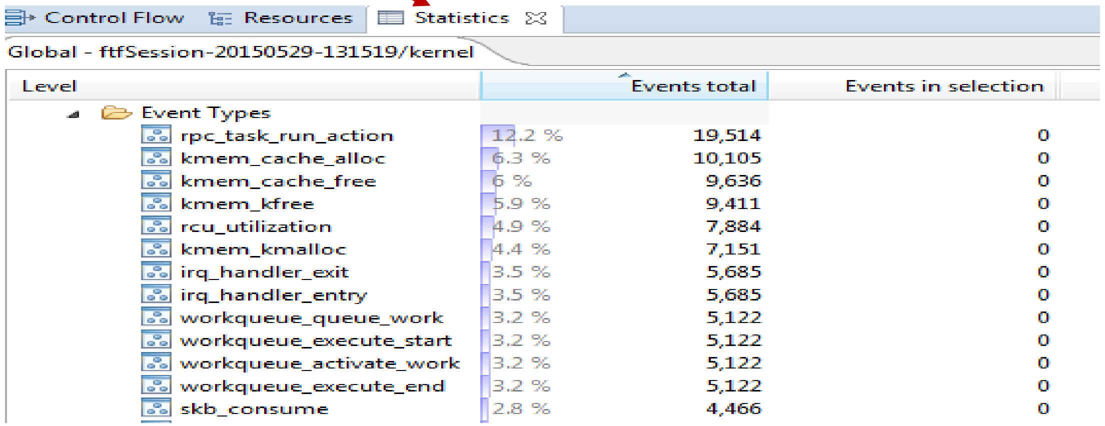
Timestamp	Channel	Type	Content
<srch>	<srch>	<srch>	<srch>
16:18:31.412 262 880	channel0_0	kmem_cache...	call_site=0xffff8000002fd250, ptr=0xffff80832fe6e380, bytes_req=128, bytes_alloc=128, gfp_flags=3...
16:18:31.412 263 560	channel0_0	kmem_kfree	call_site=0xffff8000002f4b0c, ptr=0x0
16:18:31.412 264 440	channel0_0	writeback_dir...	name=0:15, ino=8143007, index=771
16:18:31.412 266 000	channel0_0	mm_page_all...	page=0xffff7c01cebbac40, order=0, gfp_flags=16908506, migratetype=2
16:18:31.412 268 600	channel0_0	kmem_cache...	call_site=0xffff8000002fd250, ptr=0xffff80832fe6e000, bytes_req=128, bytes_alloc=128, gfp_flags=3...
16:18:31.412 269 200	channel0_0	kmem_kfree	call_site=0xffff8000002f4b0c, ptr=0x0
16:18:31.412 270 120	channel0_0	writeback_dir...	name=0:15, ino=8143007, index=772
16:18:31.412 271 600	channel0_0	mm_page_all...	page=0xffff7c01cec0ee00, order=0, gfp_flags=16908506, migratetype=2
16:18:31.412 274 440	channel0_0	kmem_cache...	call_site=0xffff8000002fd250, ptr=0xffff80832fe6ea80, bytes_req=128, bytes_alloc=128, gfp_flags=3...
16:18:31.412 275 040	channel0_0	kmem_kfree	call_site=0xffff8000002f4b0c, ptr=0x0

Trace Compass views

Resources



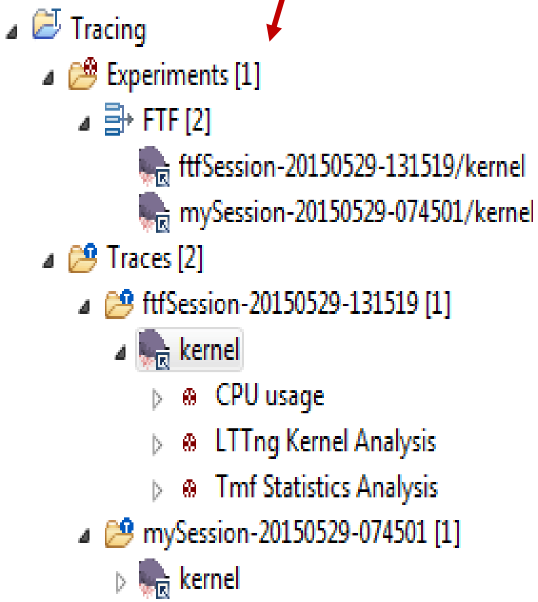
Statistics



Global - ftfSession-20150529-131519/kernel

Level	Events total		Events in selection
Event Types			
rpc_task_run_action	12.2 %	19,514	0
kmem_cache_alloc	6.3 %	10,105	0
kmem_cache_free	6 %	9,636	0
kmem_kfree	5.9 %	9,411	0
rcu_utilization	4.9 %	7,884	0
kmem_kmalloc	4.4 %	7,151	0
irq_handler_exit	3.5 %	5,685	0
irq_handler_entry	3.5 %	5,685	0
workqueue_queue_work	3.2 %	5,122	0
workqueue_execute_start	3.2 %	5,122	0
workqueue_activate_work	3.2 %	5,122	0
workqueue_execute_end	3.2 %	5,122	0
skb_consume	2.8 %	4,466	0

Project Explorer – Trace Compass session



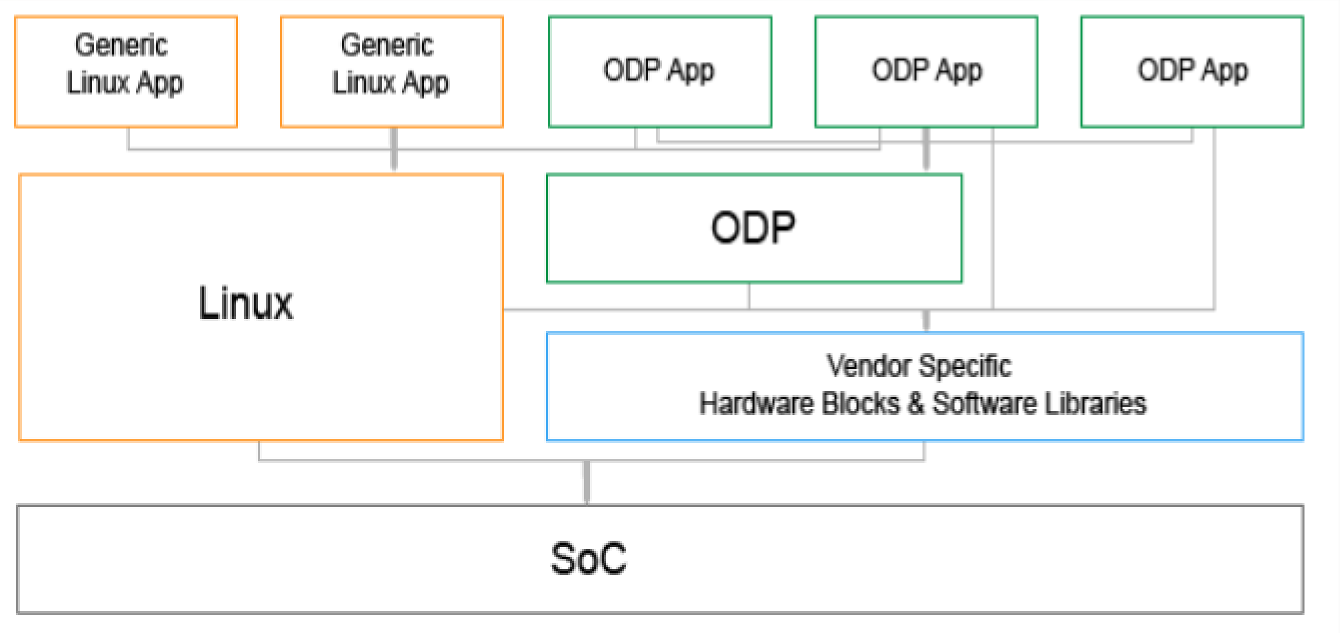
- Tracing
  - Experiments [1]
    - FTF [2]
      - ftfSession-20150529-131519/kernel
      - mySession-20150529-074501/kernel
  - Traces [2]
    - ftfSession-20150529-131519 [1]
      - kernel
        - CPU usage
        - LTTng Kernel Analysis
        - Tmf Statistics Analysis
    - mySession-20150529-074501 [1]
      - kernel

LOGGING VIA DEBUGPRINT

Introduction to ODP

What is ODP?

- The OpenDataPlane (ODP) project has been established to produce an open-source, cross-platform set of application programming interfaces (APIs) for the networking data plane
- ODP provides a data plate application programming environment that is easy to use, high performance and portable between networking SoCs



Introduction to ODP Reflector Application

It’s a sample application which performs several functions:

- Received scheduled packets are reflected back onto the same interface where the packets were originally received
- The source and destination MAC and IP addresses are swapped in received packet
- Works for all Ethernet interfaces that are defined in the resource container used by the application
- Multiple threads can be spawned for each network interface for I/O operation. In multicore environment, threads are affined with multiple cores. For single core environment, all threads are affined with the same core

Application is supported for two modes as given below:

a) Schedule PULL mode – 0 : Scheduled packets are received in PULL Mode

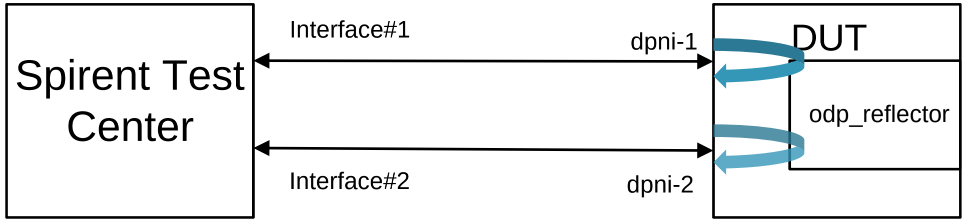
b) Schedule PUSH mode – 1 : Scheduled packets are received in PUSH Mode

Mandatory OPTIONS

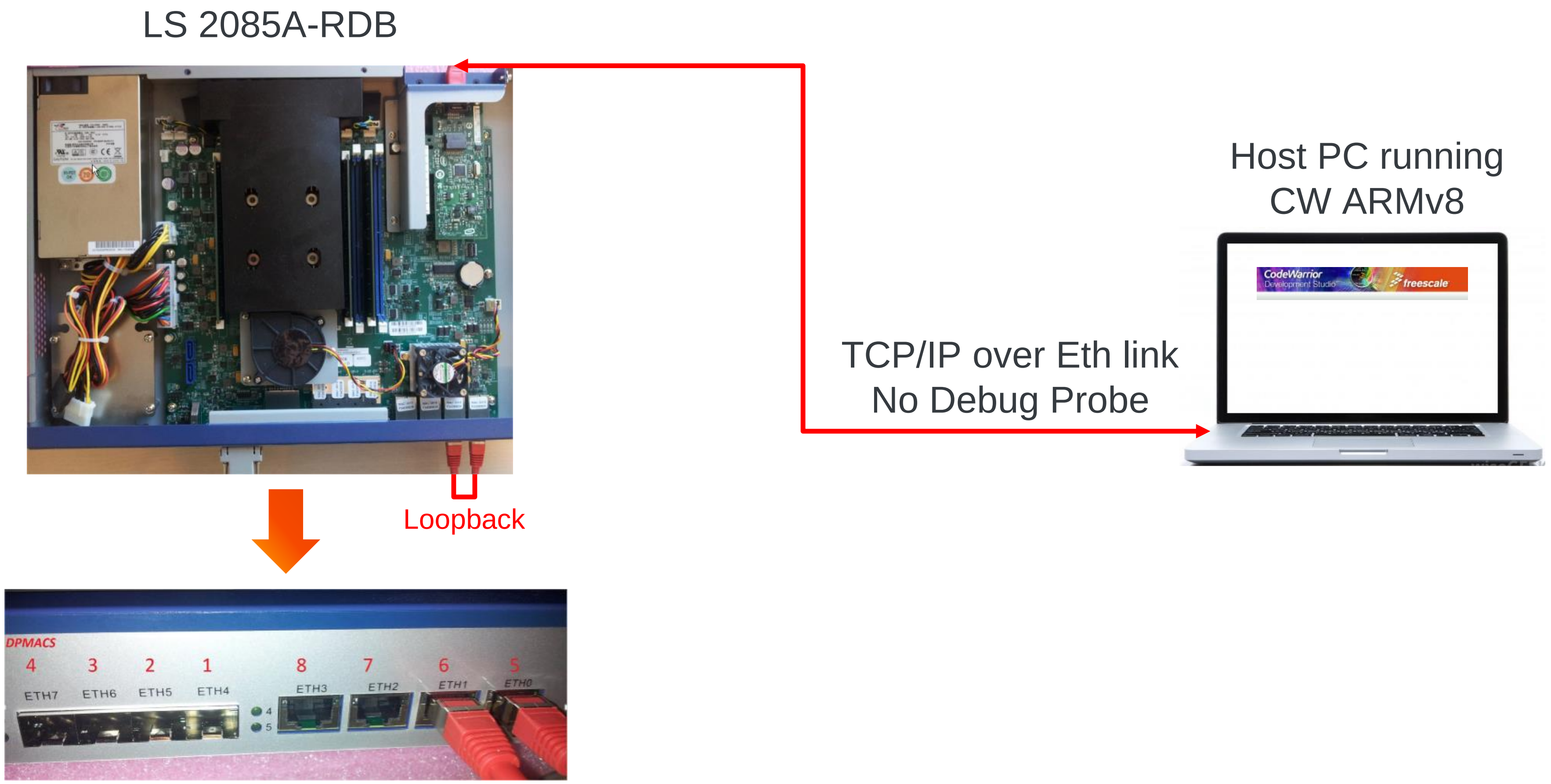
```
        -i, --interface Eth interfaces (comma-separated, no spaces)
        -m, --mode
                0 – Receive packets in Schedule PULL mode
                1 – Receive packets in Schedule PUSH mode
```

Optional OPTIONS

```
        -c, --count <number> CPU count
        -h, --help    Display help and exit
```



ODP reflector – Hardware setup using only one board



LS 2085A-RDB

Host PC running CW ARMv8

TCP/IP over Eth link
No Debug Probe

Loopback

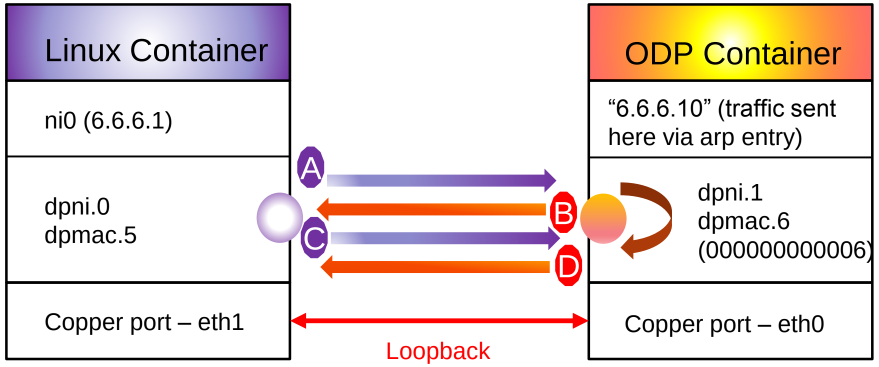
❑ For full details and steps describing the hardware and software setup please check *AN5269*

ODP reflector – software configuration

After you'll get a linux prompt, you need to issue next commands:

```
root@ls2085ardb:~# ifconfig ni0 6.6.6.1 up  1
root@ls2085ardb:~# arp -s 6.6.6.10 000000000006  2
root@ls2085ardb:~# ifconfig eth0 192.168.1.2  3
root@ls2085ardb:~# /usr/odp/scripts/dynamic_dpl.sh dpmac.6  4
…
dprc.2 Created
dpmac.6 <--------connected------> dpni.1 (00:00:00:00:0:6)
USE  dprc.2  FOR YOUR APPLICATIONS
root@ls2085ardb:~# restool dpni info dpni.0
endpoint: dpmac.5, link is up
root@ls2085ardb:~# restool dpni info dpni.1
endpoint: dpmac.6, link is down
root@ls2085ardb:~# export DPRC=dprc.2  5
root@ls2085ardb:~# /usr/odp/bin/odp_reflector -i dpni-1 -m 0 -c 8 &  6
Initializing NADK framework with following parameters:
      Resource container :dprc.2
…
setup_pkt_nadk 55-NOTICE-port =>  dpni-1 being created
setup_pkt_nadk 66-NOTICE-setup FQ 0
Port dpni-1 = Mac 00.00.00.00.00.06
<enter>
root@ls2085ardb:~# tcpdump -i ni0 &  7
<enter>
root@ls2085ardb:~# ping 6.6.6.10 -c 1  8
13:40:10.060171 IP 6.6.6.1 > 6.6.6.10: ICMP echo request, id 1953, seq 1, length 64  A
13:40:10.060207 IP 6.6.6.10 > 6.6.6.1: ICMP echo request, id 1953, seq 1, length 64  B
13:40:10.060229 IP 6.6.6.1 > 6.6.6.10: ICMP echo reply, id 1953, seq 1, length 64  C
13:40:10.060247 IP 6.6.6.10 > 6.6.6.1: ICMP echo reply, id 1953, seq 1, length 64  D
```

Auxiliary steps



1. Set ip to ni0 interface used for Linux Container
2. Add arp entry – all traffic to 6.6.6.10 will be redirect to dpni1 (which dmpac.6 – 000000000006)
3. Set ip to eth0 interface used by communication with CW
4. Allocate a new dpni (dpni.1) to dpmac.6 using restool via dynamic_dpl.sh utility script
5. Set the ODP container
6. Start the odp_reflector on dpni-1 in PULL mode
7. Start tcpdump to inspect the reflected traffic
8. Start the traffic using a single ping packet
   A. Req packet from Linux Container to ODP Container/Reflector
   B. The reflector will reflect back the same req packet swapping the IP src/dst
   C. For the above received req packet, the Linux Container will send an echo reply packet
   D. The linux networking stack sends the echo reply for the first req packet (A)

Debug Print – Fundamentals

Debug Print provides an easy method for checking Kernel & Application activities.

Debug Print consists in:

- **Server** side: running on target Linux OS for collecting Kernel Ring Buffer logs and application messages to standard output;
- **Client** side: running under CW for getting data out of the server, display and various

Linux OS running on LS 2085A-RDB

Server side

ls.target.server – reads logs and sends log data over TCP/IP

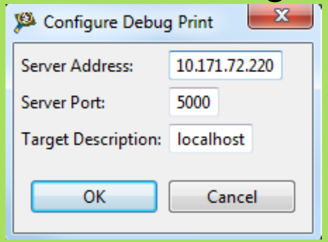
libls.linux.debugprint.lib.so – redirect user space application messages to **ls.target.server**

TCP/IP over Eth

No Debug Probe

CW ARMv8 running on Host PC

Client for **ls.target.server**



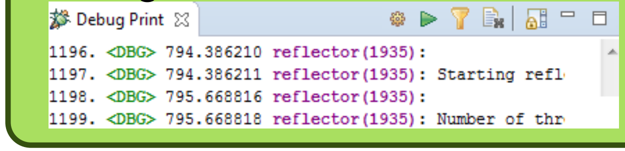
Configure Debug Print

Server Address: 10.171.72.220

Server Port: 5000

Target Description: localhost

Debug Print Viewer

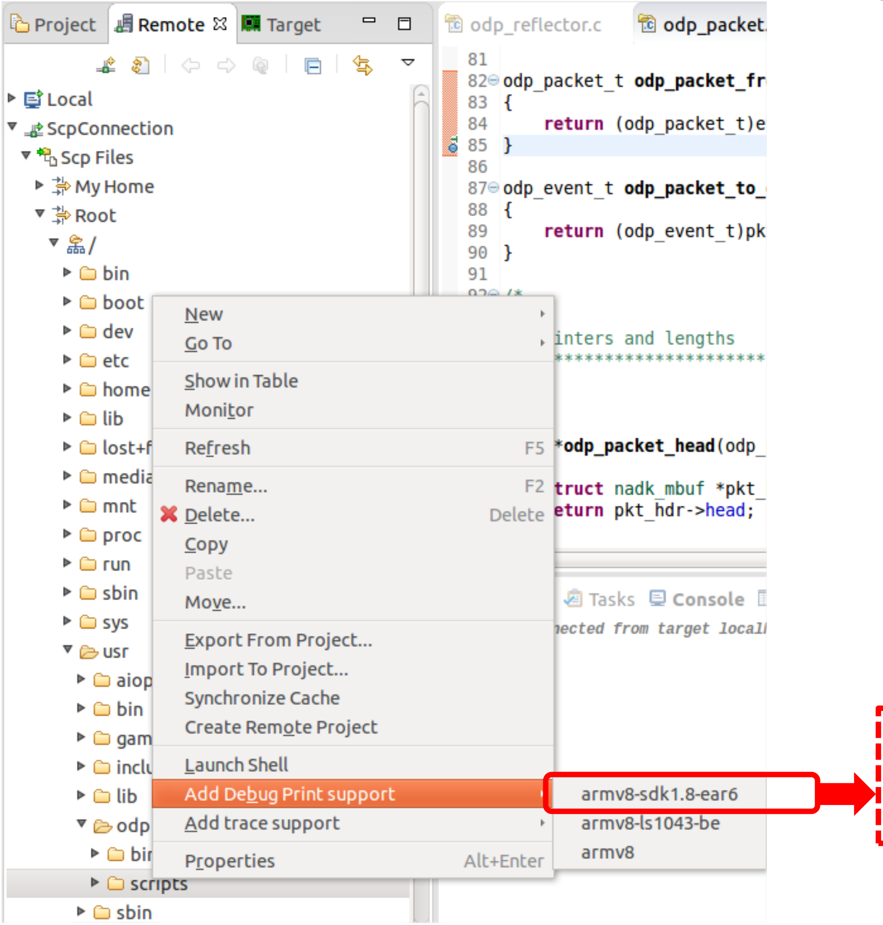


```
1196. <DBG> 794.386210 reflector(1935):
1197. <DBG> 794.386211 reflector(1935): Starting refl
1198. <DBG> 795.668816 reflector(1935):
1199. <DBG> 795.668818 reflector(1935): Number of thr
```

ACTIVITY

Server Side: setup

Using RSE all Debug Print Server utilities can be copied directly into the target Linux OS via “`Add Debug Print Support`” (scp connection-> /usr/odp/scripts)



```
odp_packet_t odp_packet_fr
{
        return (odp_packet_t)e
}

odp_event_t odp_packet_to_
{
        return (odp_event_t)pk
}

/*
inters and lengths
*********************

*odp_packet_head(odp_
truct nadk_mbuf *pkt_
eturn pkt_hdr->head;
```

```
ected from target locali
```

`linux.armv8.debugprint`
folder is created with all the prerequisites

Server Side: setup (cont’d)

Starting the server: `ls.target.server [PORT] [-k]`

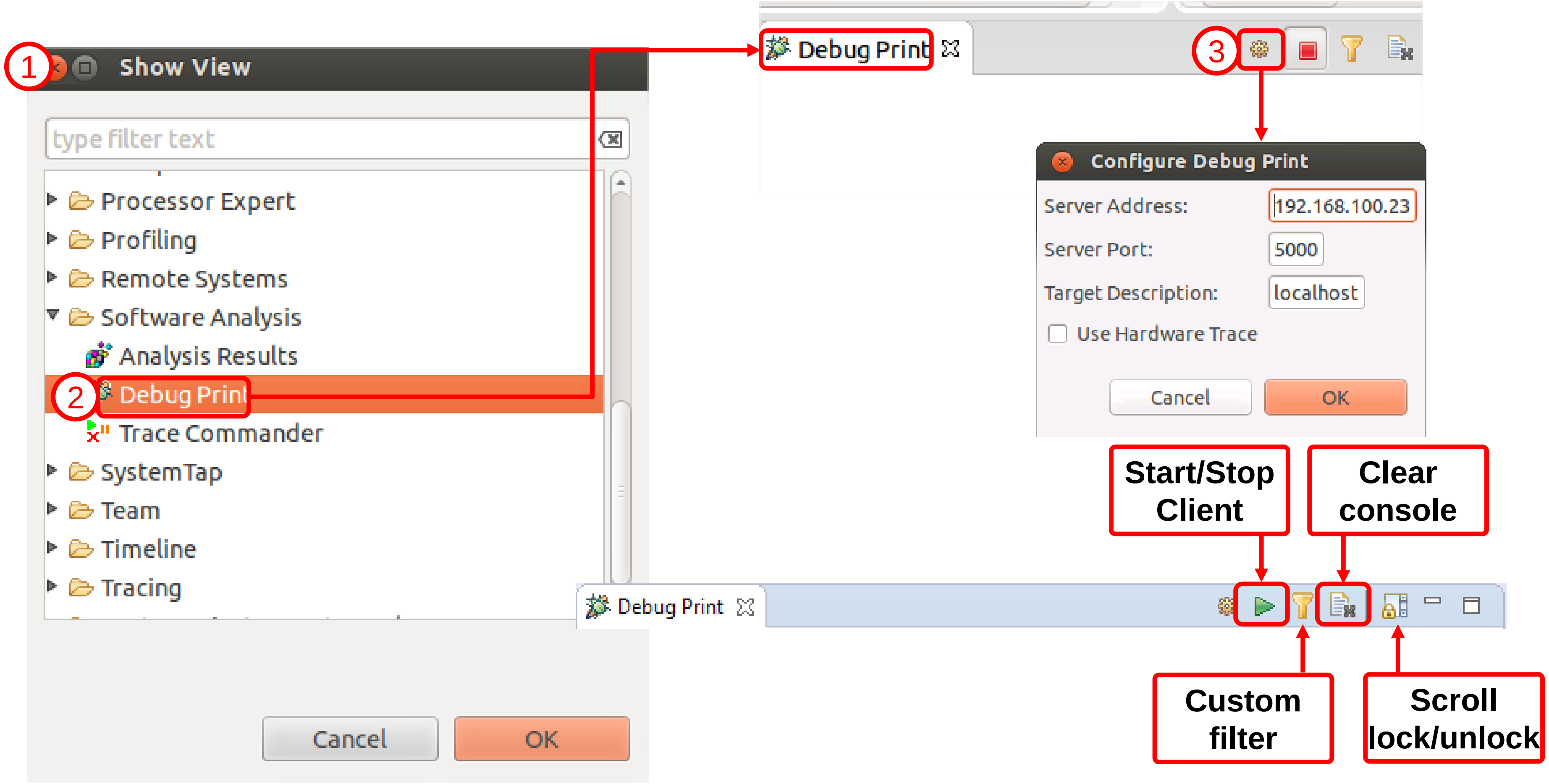
```
PORT    : default 5000
-k      : it does not clear the kernel buffer, but uses an
        internal server logic for determining which are the
        newer messages
```

E.g.: starting the Debug Print server with default settings

```
root@ls2085ardb:/usr/odp/scripts#./linux.armv8.debugprint/bin/ls.target.server &
Using port 5000
Using Kernel Ring Buffer
Initializing
```

Client Side: setup

Open the `Debug Print Viewer` and connect the Client with the Server using TCP/IP and Port



Server Address: 192.168.100.23

Server Port: 5000

Target Description: localhost

☐ Use Hardware Trace

Start/Stop Client

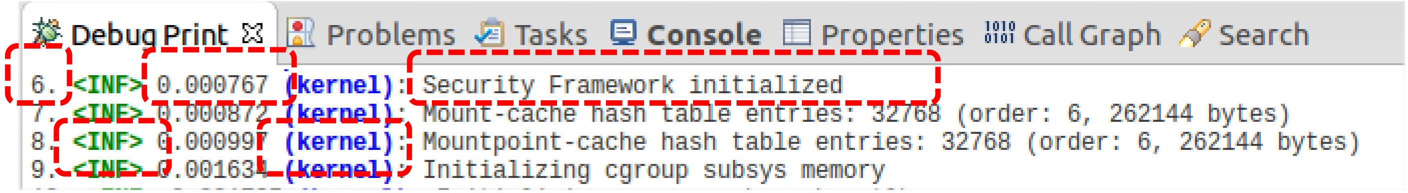
Clear console

Custom filter

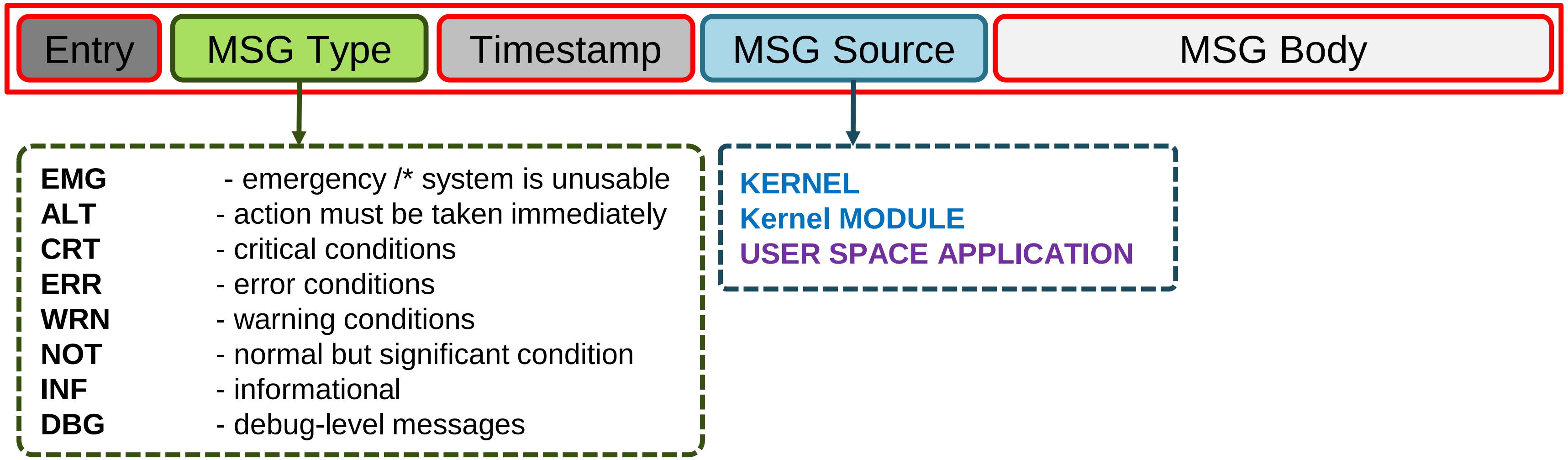
Scroll lock/unlock

Client Side: Debug Print messages format

Entry format:



```
6. <INF> 0.000767 (kernel): Security Framework initialized
7. <INF> 0.000872 (kernel): Mount-cache hash table entries: 32768 (order: 6, 262144 bytes)
8. <INF> 0.000997 (kernel): Mountpoint-cache hash table entries: 32768 (order: 6, 262144 bytes)
9. <INF> 0.001634 (kernel): Initializing cgroup subsys memory
```



EMG	- emergency /* system is unusable
ALT	- action must be taken immediately
CRT	- critical conditions
ERR	- error conditions
WRN	- warning conditions
NOT	- normal but significant condition
INF	- informational
DBG	- debug-level messages

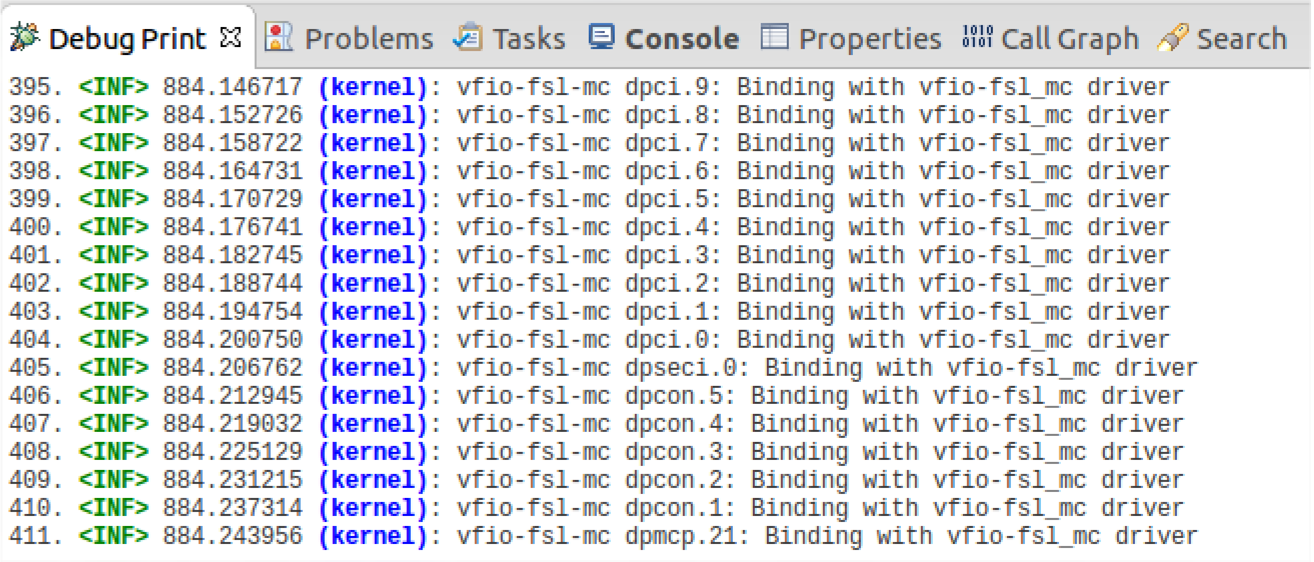
KERNEL
Kernel MODULE
USER SPACE APPLICATION

Reflector: startup

Step 1: setup the environment according with reflector requirements:

```
root@ls2085ardb:/usr/odp/scripts/dynamic_dpl.sh dpmac.6
dpcon.1 assigned to  dprc.2
dpcon.2 assigned to  dprc.2
dpcon.3 assigned to  dprc.2
dpcon.4 assigned to  dprc.2
dpcon.5 assigned to  dprc.2
dpseci.0 assigned to  dprc.2
```

Step 2: Debug Print client will catch all logs during setup:



```
395. <INF> 884.146717 (kernel): vfio-fsl-mc dpci.9: Binding with vfio-fsl_mc driver
396. <INF> 884.152726 (kernel): vfio-fsl-mc dpci.8: Binding with vfio-fsl_mc driver
397. <INF> 884.158722 (kernel): vfio-fsl-mc dpci.7: Binding with vfio-fsl_mc driver
398. <INF> 884.164731 (kernel): vfio-fsl-mc dpci.6: Binding with vfio-fsl_mc driver
399. <INF> 884.170729 (kernel): vfio-fsl-mc dpci.5: Binding with vfio-fsl_mc driver
400. <INF> 884.176741 (kernel): vfio-fsl-mc dpci.4: Binding with vfio-fsl_mc driver
401. <INF> 884.182745 (kernel): vfio-fsl-mc dpci.3: Binding with vfio-fsl_mc driver
402. <INF> 884.188744 (kernel): vfio-fsl-mc dpci.2: Binding with vfio-fsl_mc driver
403. <INF> 884.194754 (kernel): vfio-fsl-mc dpci.1: Binding with vfio-fsl_mc driver
404. <INF> 884.200750 (kernel): vfio-fsl-mc dpci.0: Binding with vfio-fsl_mc driver
405. <INF> 884.206762 (kernel): vfio-fsl-mc dpseci.0: Binding with vfio-fsl_mc driver
406. <INF> 884.212945 (kernel): vfio-fsl-mc dpcon.5: Binding with vfio-fsl_mc driver
407. <INF> 884.219032 (kernel): vfio-fsl-mc dpcon.4: Binding with vfio-fsl_mc driver
408. <INF> 884.225129 (kernel): vfio-fsl-mc dpcon.3: Binding with vfio-fsl_mc driver
409. <INF> 884.231215 (kernel): vfio-fsl-mc dpcon.2: Binding with vfio-fsl_mc driver
410. <INF> 884.237314 (kernel): vfio-fsl-mc dpcon.1: Binding with vfio-fsl_mc driver
411. <INF> 884.243956 (kernel): vfio-fsl-mc dpmcp.21: Binding with vfio-fsl_mc driver
```

Reflector: startup (cont'd)

STEP3: redirect reflector standard output to Server by loading the appropriate library and start the application:

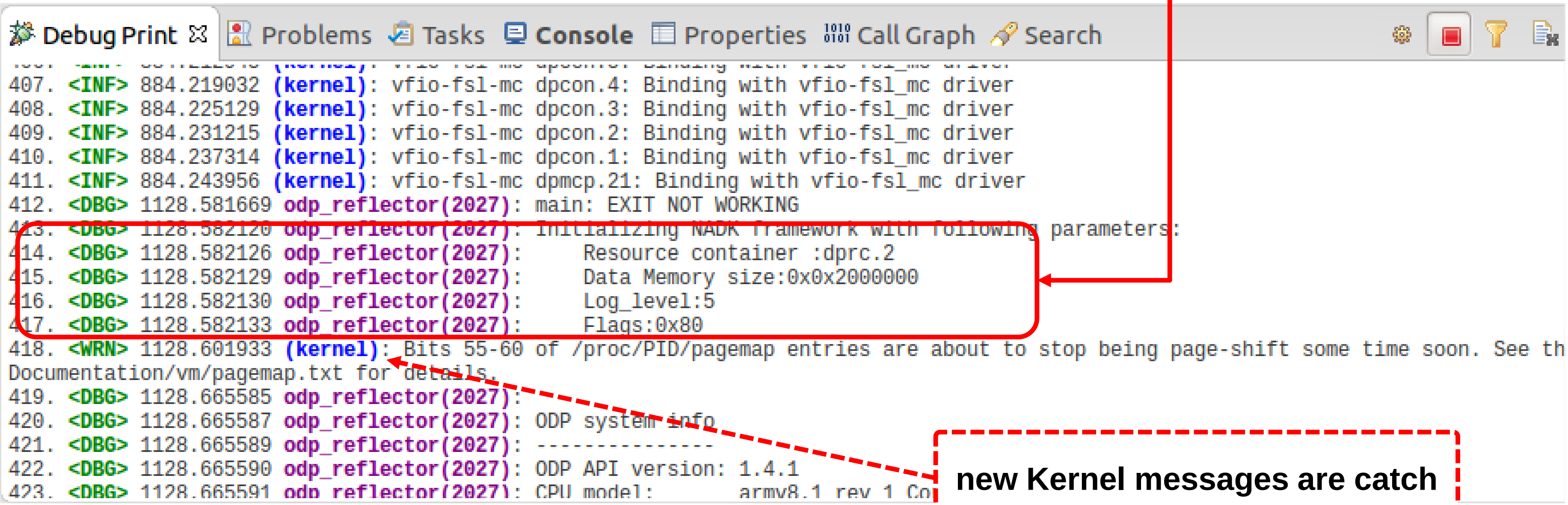
```
root@ls2085ardb:/usr/odp/bin/# export DPRC=dprc.2
```

```
root@ls2085ardb:/usr/odp/bin/#
LD_PRELOAD=/usr/odp/scripts/linux.armv8.debugprint/lib/libls.linux.debugprint.so.1.0
/usr/odp/bin/odp_reflector -i dpni-1 -m 0 -c 8
```

At this point the Linux console should look like:

```
odp_nadk_scan_device_list 192-NOTICE-dpconc-2 being created
odp_nadk_scan_device_list 192-NOTICE-dpconc-1 being
createdodp_schedule.c:160:odp_schedule_init_global():Schedule init ...
odp_schedule.c:214:odp_schedule_init_global():done

odp_nadk_scan_device_list 192-NOTICE-dpni-1 being
createdodp_crypto.c:1153:odp_crypto_init_global():Crypto init ...
odp_pool.c:236:odp_pool_create():Configuring buffer pool list
0x32a05c00odp_pool.c:368:odp_pool_print():NADK BMAN buffer pool bpid
4odp_packet_io.c:239:odp_pktio_open():Allocating nadk pktio
odp_packet_nadk.c:45:setup_pkt_nadk():setup_pkt_nadk

setup_pkt_nadk 55-NOTICE-port =>  dpni-1 being created
setup_pkt_nadk 66-NOTICE-setup FQ 0odp_schedule.c:345:odp_schedule_queue():setup VQ 0
with handle 0x41
```

Reflector: Debug Print results



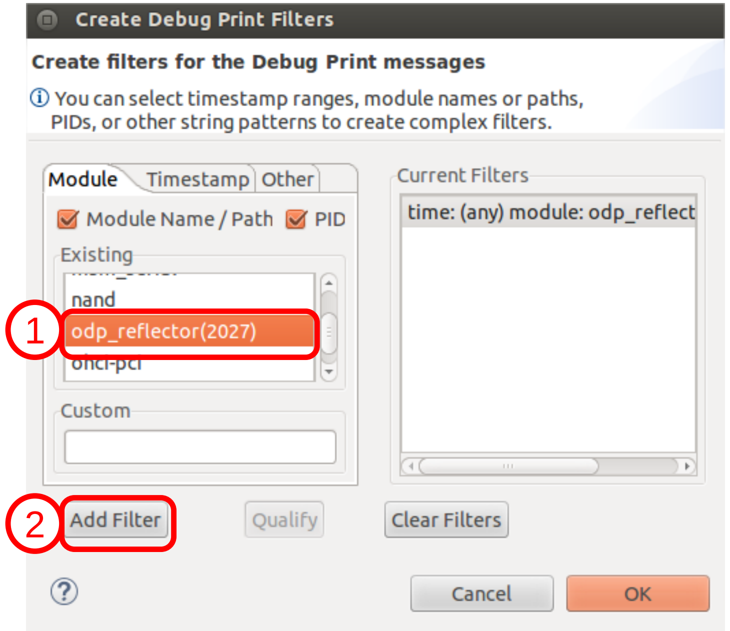
```
407. <INF> 884.219032 (kernel): vfio-fsl-mc dpcon.4: Binding with vfio-fsl_mc driver
408. <INF> 884.225129 (kernel): vfio-fsl-mc dpcon.3: Binding with vfio-fsl_mc driver
409. <INF> 884.231215 (kernel): vfio-fsl-mc dpcon.2: Binding with vfio-fsl_mc driver
410. <INF> 884.237314 (kernel): vfio-fsl-mc dpcon.1: Binding with vfio-fsl_mc driver
411. <INF> 884.243956 (kernel): vfio-fsl-mc dpmcp.21: Binding with vfio-fsl_mc driver
412. <DBG> 1128.581669 odp_reflector(2027): main: EXIT NOT WORKING
413. <DBG> 1128.582120 odp_reflector(2027): Initializing NADK framework with following parameters:
414. <DBG> 1128.582126 odp_reflector(2027):     Resource container :dprc.2
415. <DBG> 1128.582129 odp_reflector(2027):     Data Memory size:0x0x2000000
416. <DBG> 1128.582130 odp_reflector(2027):     Log_level:5
417. <DBG> 1128.582133 odp_reflector(2027):     Flags:0x80
418. <WRN> 1128.601933 (kernel): Bits 55-60 of /proc/PID/pagemap entries are about to stop being page-shift some time soon. See th
Documentation/vm/pagemap.txt for details.
419. <DBG> 1128.665585 odp_reflector(2027):
420. <DBG> 1128.665587 odp_reflector(2027): ODP system info
421. <DBG> 1128.665589 odp_reflector(2027): ---------------
422. <DBG> 1128.665590 odp_reflector(2027): ODP API version: 1.4.1
423. <DBG> 1128.665591 odp_reflector(2027): CPU model:       armv8.1 rev 1 Co
```

Reflector application messages

new Kernel messages are catch

Reflector: Debug Print results (cont’d)

Customize the Debug Print Client to display only relevant information: messages for reflector



Create Debug Print Filters

Create filters for the Debug Print messages

ⓘ You can select timestamp ranges, module names or paths, PIDs, or other string patterns to create complex filters.

Module | Timestamp | Other

☑ Module Name / Path ☑ PID

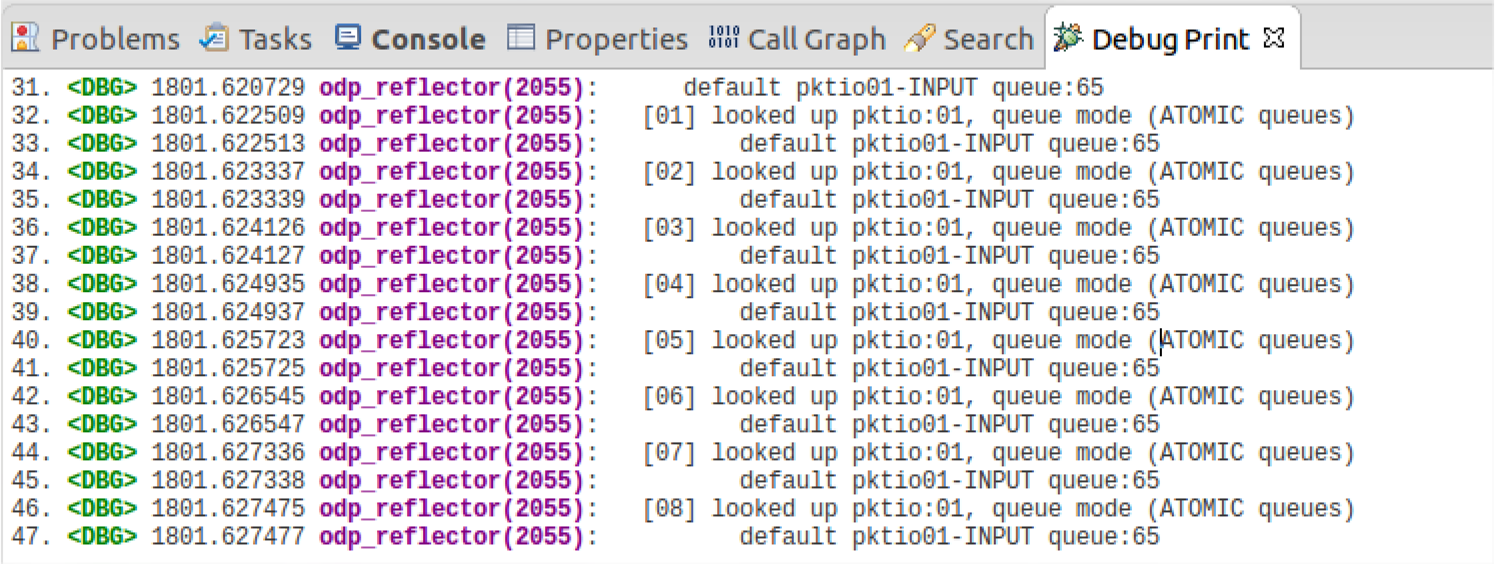
Existing

nand
odp_reflector(2027)
ohci-pci

Custom

Current Filters

time: (any) module: odp_reflect



```
<DBG> 1801.620729 odp_reflector(2055):      default pktio01-INPUT queue:65
<DBG> 1801.622509 odp_reflector(2055):   [01] looked up pktio:01, queue mode (ATOMIC queues)
<DBG> 1801.622513 odp_reflector(2055):        default pktio01-INPUT queue:65
<DBG> 1801.623337 odp_reflector(2055):   [02] looked up pktio:01, queue mode (ATOMIC queues)
<DBG> 1801.623339 odp_reflector(2055):        default pktio01-INPUT queue:65
<DBG> 1801.624126 odp_reflector(2055):   [03] looked up pktio:01, queue mode (ATOMIC queues)
<DBG> 1801.624127 odp_reflector(2055):        default pktio01-INPUT queue:65
<DBG> 1801.624935 odp_reflector(2055):   [04] looked up pktio:01, queue mode (ATOMIC queues)
<DBG> 1801.624937 odp_reflector(2055):        default pktio01-INPUT queue:65
<DBG> 1801.625723 odp_reflector(2055):   [05] looked up pktio:01, queue mode (ATOMIC queues)
<DBG> 1801.625725 odp_reflector(2055):        default pktio01-INPUT queue:65
<DBG> 1801.626545 odp_reflector(2055):   [06] looked up pktio:01, queue mode (ATOMIC queues)
<DBG> 1801.626547 odp_reflector(2055):        default pktio01-INPUT queue:65
<DBG> 1801.627336 odp_reflector(2055):   [07] looked up pktio:01, queue mode (ATOMIC queues)
<DBG> 1801.627338 odp_reflector(2055):        default pktio01-INPUT queue:65
<DBG> 1801.627475 odp_reflector(2055):   [08] looked up pktio:01, queue mode (ATOMIC queues)
<DBG> 1801.627477 odp_reflector(2055):        default pktio01-INPUT queue:65
```

Debug Print Considerations

- Debug Print Client can show up messages from Kernel, Modules and User Applications in a easy straightforward fashion allowing filtering based on source/timestamps/keywords

- Attaching like use cases to a running application is not supported since the Debug Print redirect library must be loaded before application is getting started

- Debug Print Server and Client can be started at any time

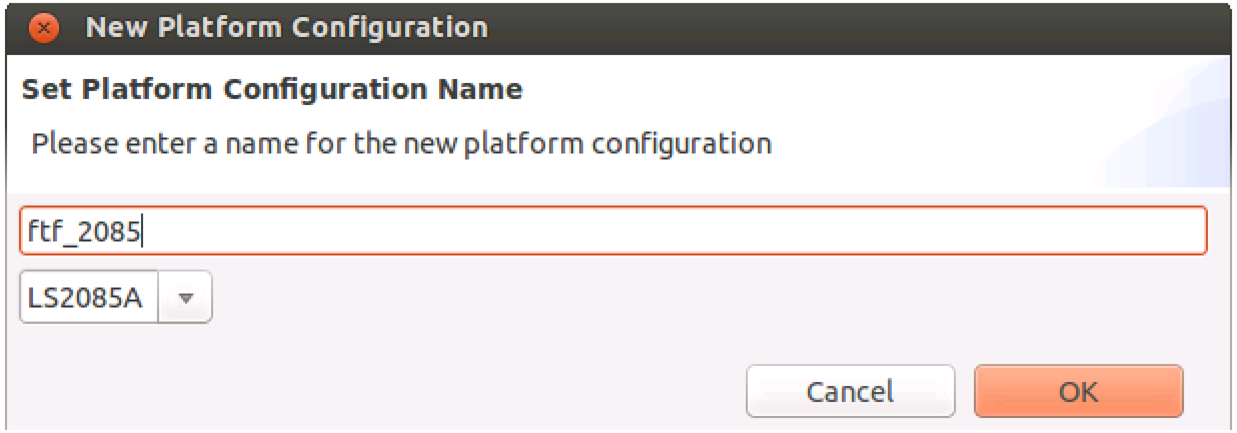
U-BOOT TRACING

U-Boot tracing

- Perform trace on u-boot execution
- Catch u-boot stages, before and after code relocation
- No hassle for users with trace buffer for each stage
- Integration with Debugger for proper injection of trace settings when code relocation is done

U-Boot tracing – setting Trace configuration

1. Open “Trace and Profile” tab. Make a new platform configuration – make sure that you choose the right platform architecture -> LS2085A



New Platform Configuration

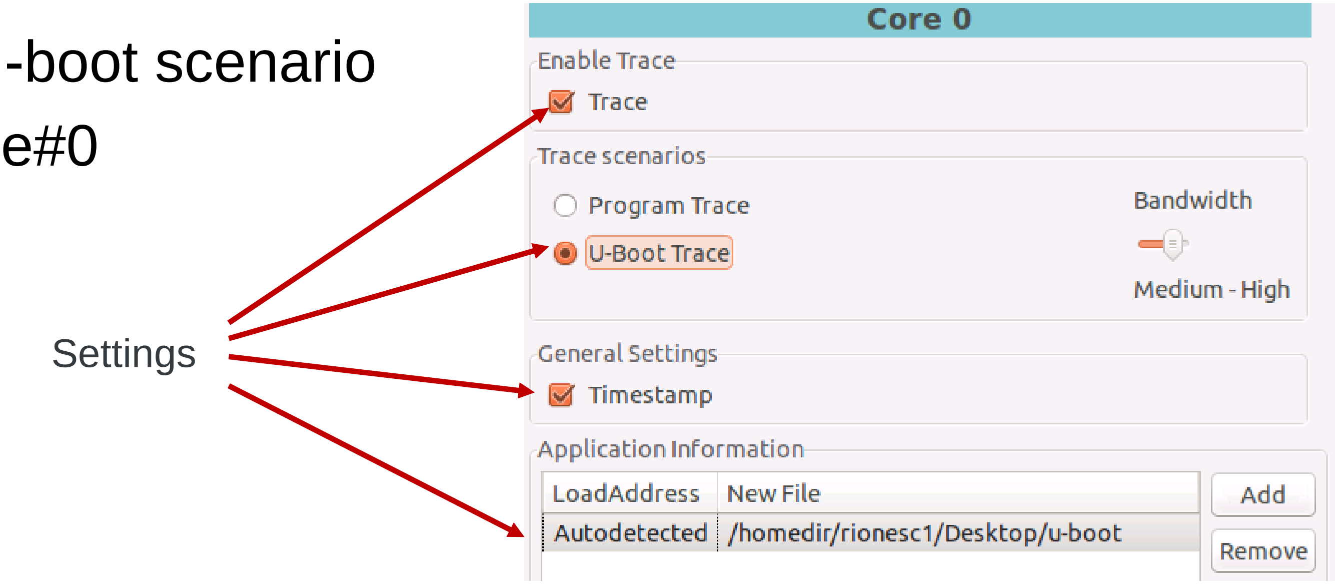
Set Platform Configuration Name

Please enter a name for the new platform configuration

ftf_2085

LS2085A

2. Enable only the Core#0 trace
3. Enable Timestamp and U-boot scenario
4. Add u-boot binary for Core#0



Settings

Core 0

Enable Trace

☑ Trace

Trace scenarios

○ Program Trace

● U-Boot Trace

Bandwidth: Medium - High

General Settings

☑ Timestamp

Application Information

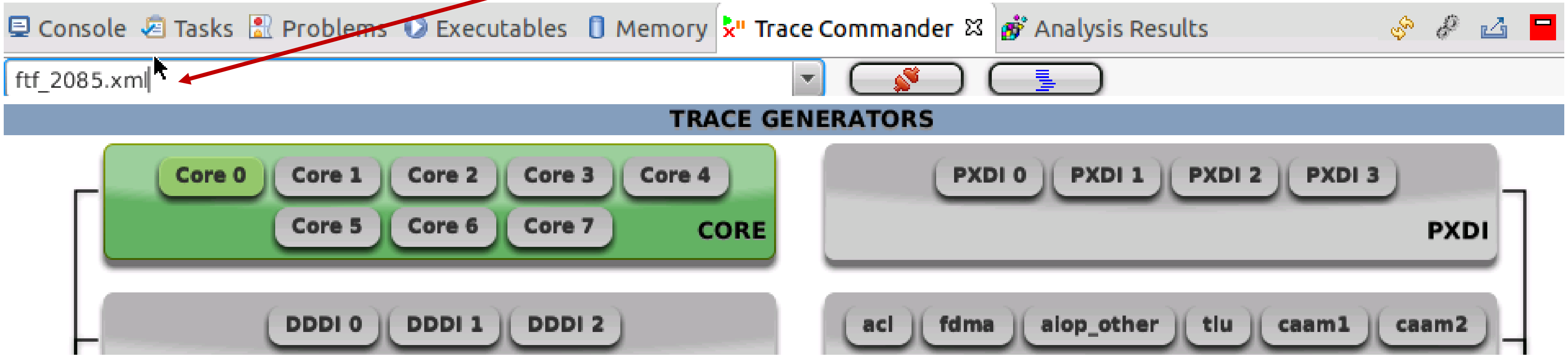
LoadAddress	New File
Autodetected	/homedir/rionesc1/Desktop/u-boot

U-Boot tracing – launching u-boot project

5. Press on “*Debug*” button to launch the project
6. Target will stop at address 0x00000000
7. Open *Trace Commander* view (Window -> Show View -> Other -> Software Analysis)

Make sure that the right platform config is used:

Platform config



ftf_2085.xml

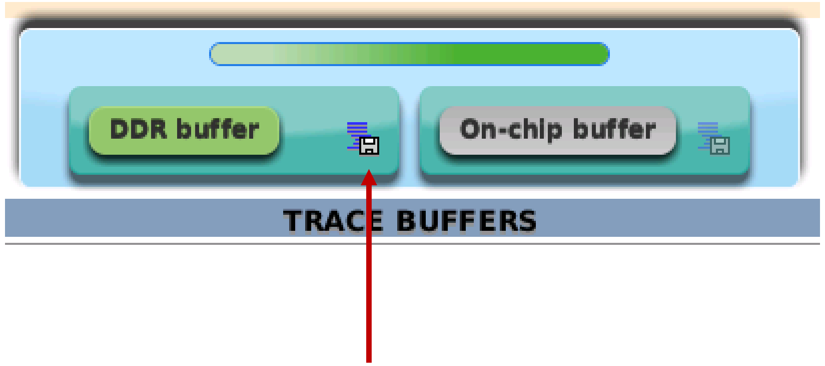
U-Boot tracing – running u-boot project

8. Press on *Connect* button to apply trace settings on target

9. After connection is done, resume execution of target (you may notice into a terminal linked with target serial connection)

10. After u-boot was executed (see in terminal when u-boot is done), suspend target execution from CodeWarrior

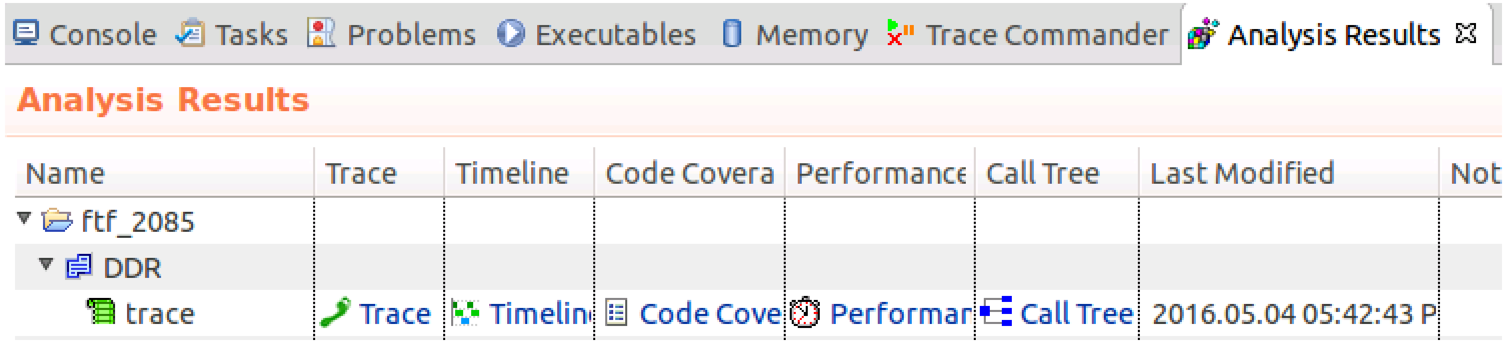
11. Collect trace from Trace Commander view



Collect trace by pressing this button

U-Boot tracing – see results

12. Open *Analysis Results* view and refresh it to display the new collected data



Analysis Results

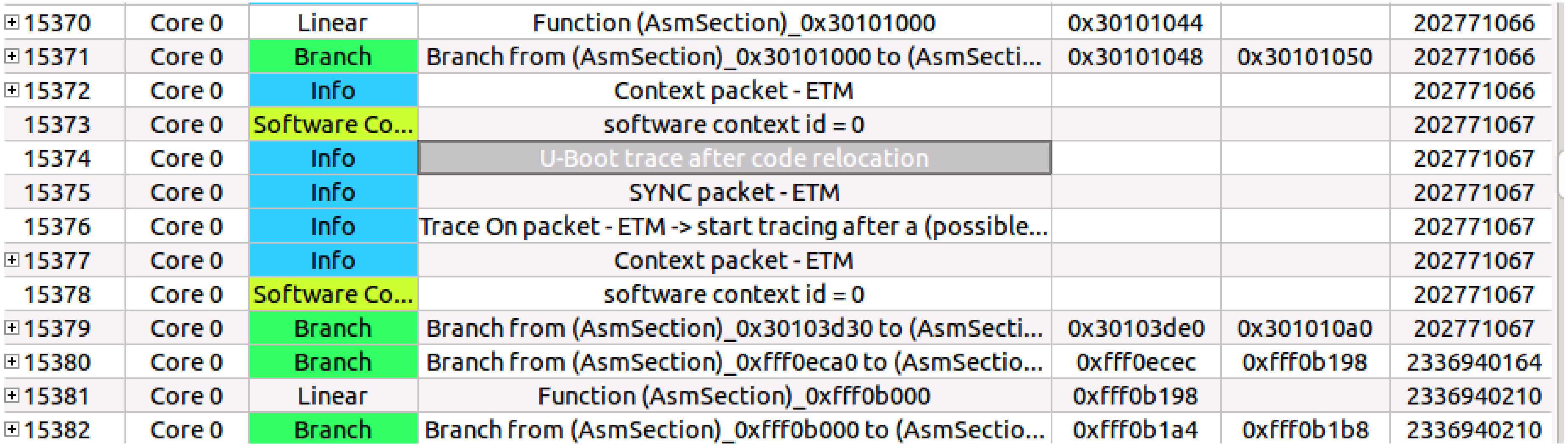
Name	Trace	Timeline	Code Covera	Performance	Call Tree	Last Modified	Not
ftf_2085							
DDR							
trace	Trace	Timelin	Code Cove	Performar	Call Tree	2016.05.04 05:42:43 P	

13. Open Trace, by clicking on *Trace* link.

14. Search for “U-Boot trace before code relocation” and “U-Boot trace after code relocation”

U-Boot tracing – see results

15. Notice the change in timestamp and address of execution after code relocation.



⊞ 15370	Core 0	Linear	Function (AsmSection)_0x30101000	0x30101044		202771066
⊞ 15371	Core 0	Branch	Branch from (AsmSection)_0x30101000 to (AsmSecti...	0x30101048	0x30101050	202771066
⊞ 15372	Core 0	Info	Context packet - ETM			202771066
15373	Core 0	Software Co...	software context id = 0			202771067
15374	Core 0	Info	U-Boot trace after code relocation			202771067
15375	Core 0	Info	SYNC packet - ETM			202771067
15376	Core 0	Info	Trace On packet - ETM -> start tracing after a (possible...			202771067
⊞ 15377	Core 0	Info	Context packet - ETM			202771067
15378	Core 0	Software Co...	software context id = 0			202771067
⊞ 15379	Core 0	Branch	Branch from (AsmSection)_0x30103d30 to (AsmSecti...	0x30103de0	0x301010a0	202771067
⊞ 15380	Core 0	Branch	Branch from (AsmSection)_0xfff0eca0 to (AsmSectio...	0xfff0ecec	0xfff0b198	2336940164
⊞ 15381	Core 0	Linear	Function (AsmSection)_0xfff0b000	0xfff0b198		2336940210
⊞ 15382	Core 0	Branch	Branch from (AsmSection)_0xfff0b000 to (AsmSectio...	0xfff0b1a4	0xfff0b1b8	2336940210

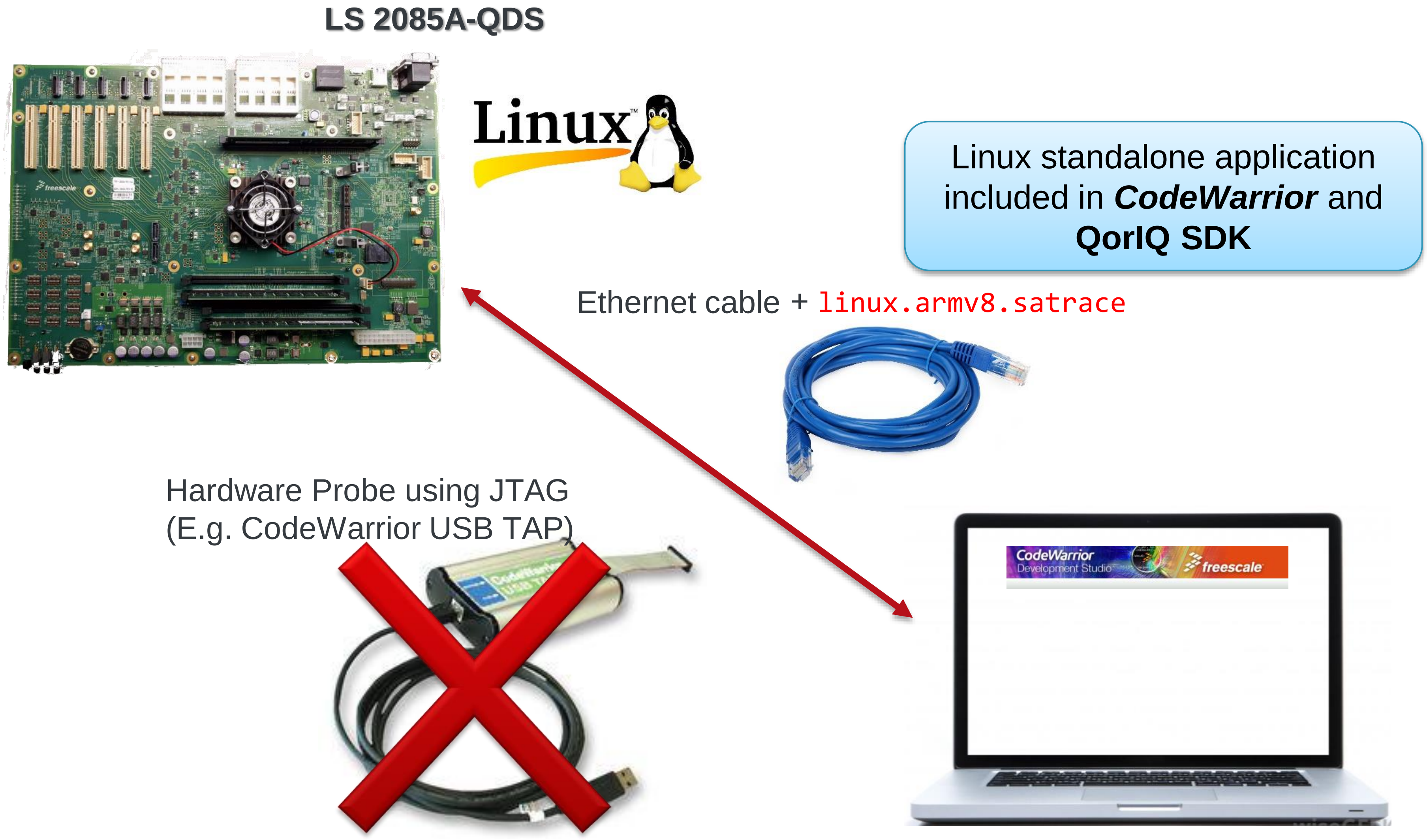
LINUX USER APPLICATION TRACING

Linux Probe-less Trace

- Based on a software probe
  - Linux cross-compiled application
  - CW and SDK component

- Advantages
  - Speed
    - contains only what is needed
  - Speed
    - all services are hosted on target machine
  - Nonintrusive
    - no need to instrument the target application
  - Simple API
    - can be effortlessly integrated into any testing framework
  - Data-driven
    - the configurator and probe can be easily tuned up using *xml* files

Linux Probe-less Trace – Hardware setup



LS 2085A-QDS

Linux standalone application included in ***CodeWarrior*** and **QorIQ SDK**

Ethernet cable + `linux.armv8.satrace`

Hardware Probe using JTAG
(E.g. CodeWarrior USB TAP)

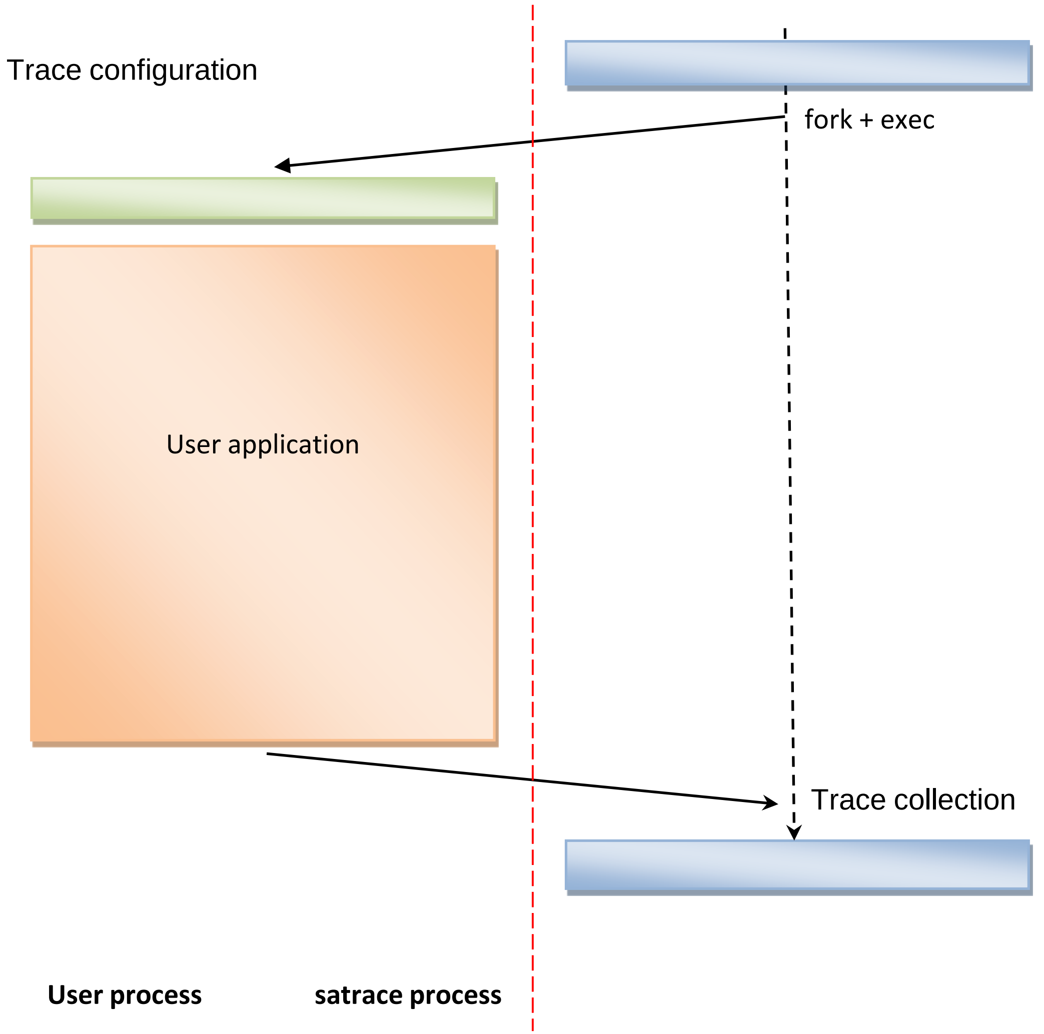
Linux probeless trace – Command Line API

```
This application starts and collects trace on target.

Usage: ./linux.armv8-sdk1.8-ear6.satrace/bin/ls.linux.satrace [Options] app [app_args]

User space options:
  -A [ --archive-file ] arg (=[app_name].cwzsa)
                                        Archive path
  -b [ --backtrace ]    Shows backtrace on SEGFAULT.

        [app_name] - Name of the traced application.
Common options:
  -T [ --multithreading ]  Enables multithreading support.
  -p [ --pid ] PID         Attach to a process giving a PID.
  --vmid vmid              Virtual machine ID
  --start-trace address    Start tracepoint
  --stop-trace address     Stop tracepoint
  --include-range range    Include range
  --exclude-range range    Exclude range

Kernel space options:
  -K [ --kernel ] path  Archive path.
  -i [ --kernel-image ] path vmlinux image compiled with debugging symbols.

System trace options:
  -S [ --system ] arg (=[app_name].scwzsa)
                                        Archive path.
  -i [ --kernel-image ] path vmlinux image compiled with debugging symbols.
  -b [ --backtrace ]    Shows backtrace on SEGFAULT.

General options:
  -v [ --verbose ]          Verbose mode
  -V [ --version ]          Product version
  -h [ --help ]             Displays this help message
  -c [ --config-file ] path Configuration file
  --soc arg (=LS2085A)      Name of the SoC
```

```
Notes:
    Do not mix kernel and user space options, otherwise all user space options will
    be ignored.
    The kernel space trace will be collected after catching the SIGINT signal
    (CTRL+C).
    -A will create an archive with a custom name.

    [range] - An interval specified using one of the following formats :
            0x2000-0x3000                Address range [0x2000, 0x3000]
            libpthread                   Executable code from libpthread.so
            init_linuxrc                 Address range based on kernel function name
                                         Covers all instructions from init_linuxrc
            init_linuxrc-init_linuxrc+8  Includes/Excludes first 8 bytes from
                                         init_linuxrc
            ipv6.ko                      Includes/Excludes 'ipv6' kernel module
    [address] - An address specified using one of the following formats :
            0x2000                       Hex address
            libpthread+200               Offset from a shared libary (libpthread.so)
            init_linuxrc                 Address based on kernel function name
            init_linuxrc+8               Kernel function offset
            ipv6.ko                      Kernel module offset
    --vmid argument is compatible only with address range filters.
```

```
Examples :
        ls.linux.satrace -A archive.cwzsa ./my_app
        ls.linux.satrace ./my_app my_arg1 my_arg2
        ls.linux.satrace -K kernelTest
        ls.linux.satrace -K kernelTest -i ~/vmlinux
        ls.linux.satrace -p 534
        ls.linux.satrace -p 534 -A attachTrace
        ls.linux.satrace -p 534 --include-range=init_linuxrc-init_linuxrc+20
        ls.linux.satrace -p 534 --exclude-range=0x3000-0x7000
        ls.linux.satrace --vmid=1 -S arname -p 534 --include-range=0x2000-0x3000
                         --include-range=0x4000-0x5000
        ls.linux.satrace -p 534 --start-trace=__switch_to
                         --stop-trace=__switch_to+0x200
```

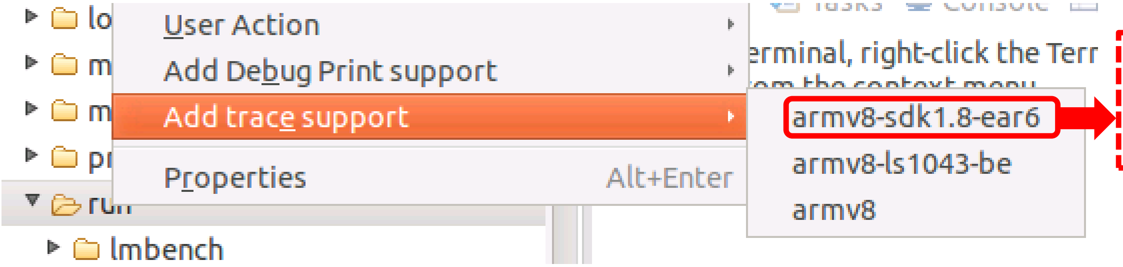
Linux user space application trace *(command line)*



ACTIVITY

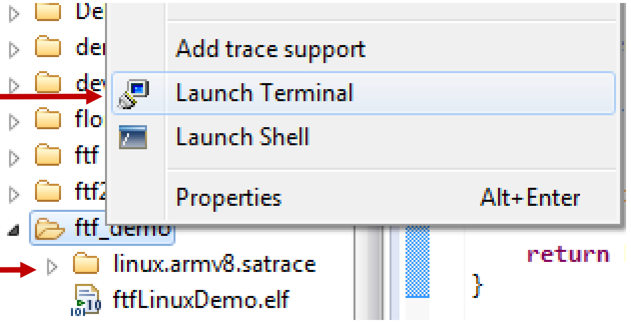
Linux probeless trace *(CodeWarrior)*

1. Add trace support (right click on your demo folder):



`linux.armv8.satrace` folder is created with all the prerequisites

2. Right-click on RSE tree → Launch a Terminal

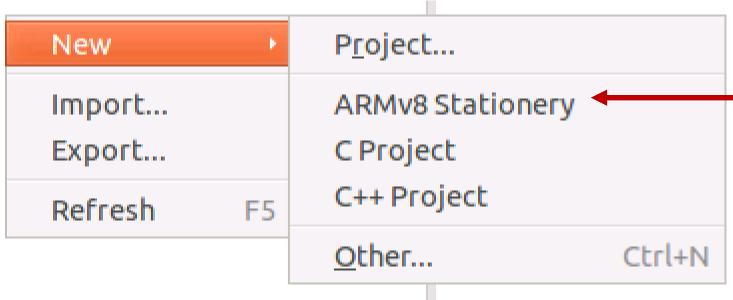


New Terminal

Added trace support

Linux probeless trace *(CodeWarrior)*

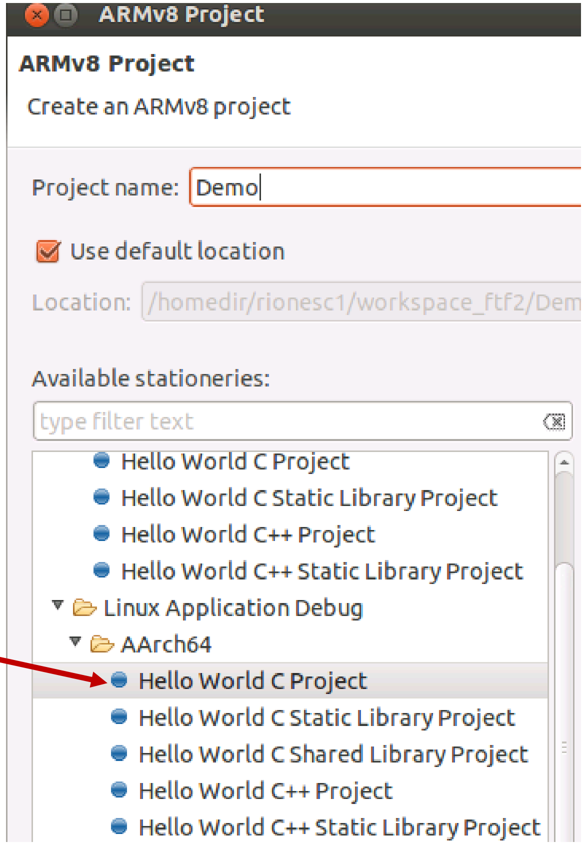
3. Open Project Explorer. Right-click and Create new project



New ARMv8 stationary project

4. Create a Linux Application Debug project. Build it.

New Linux Application Debug project



ARMv8 Project

ARMv8 Project

Create an ARMv8 project

Project name: Demo

☑ Use default location

Location: /homedir/rionesc1/workspace_ftf2/Dem

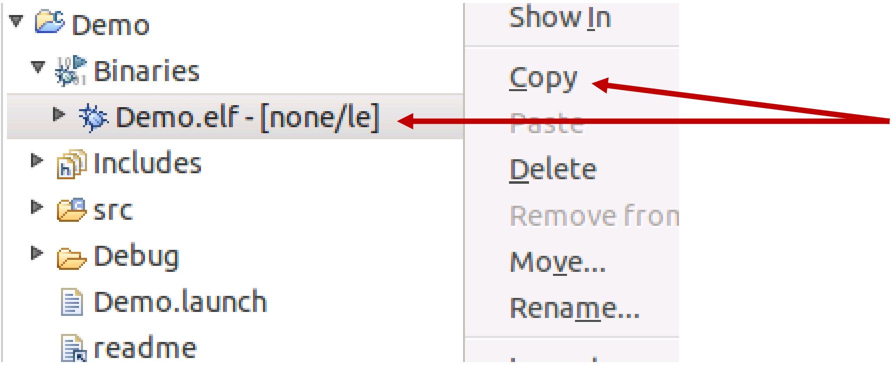
Available stationeries:

type filter text

- Hello World C Project
- Hello World C Static Library Project
- Hello World C++ Project
- Hello World C++ Static Library Project
- Linux Application Debug
  - AArch64
    - Hello World C Project
    - Hello World C Static Library Project
    - Hello World C Shared Library Project
    - Hello World C++ Project
    - Hello World C++ Static Library Project

Linux probeless trace *(CodeWarrior)*

5. Open Project Explorer. Right-click and Create new project

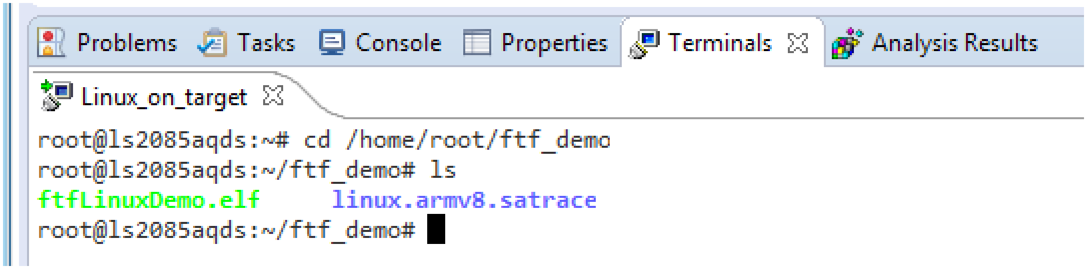


Copy elf binary

6. In Remote Systems view, copy elf file to demo folder

Linux probeless trace *(CodeWarrior)*

7. Run ***ls*** in Terminal to check the files in your folder:



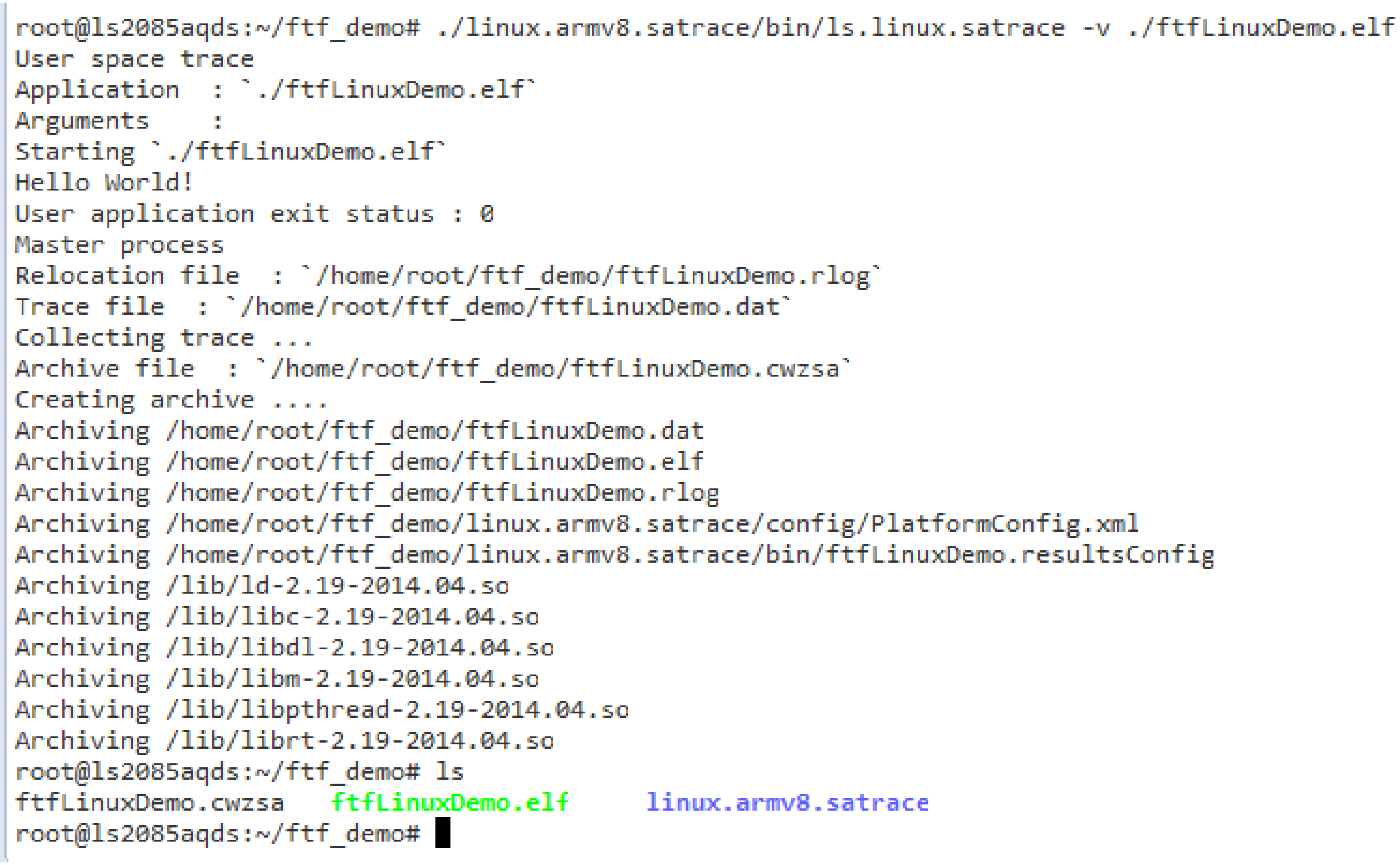
```
root@ls2085aqds:~# cd /home/root/ftf_demo
root@ls2085aqds:~/ftf_demo# ls
ftfLinuxDemo.elf     linux.armv8.satrace
root@ls2085aqds:~/ftf_demo# 
```

8. Launch application with trace support. Make sure you pass *–v* flag in order to see all stages in verbose mode:

./linux.armv8.satrace/bin/ls.linux.satrace -v ./ftfLinuxDemo.elf

Linux probeless trace *(CodeWarrior)*

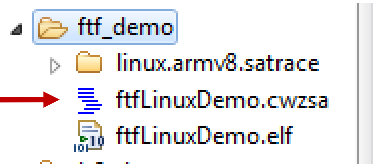
9. The results will be:



```
root@ls2085aqds:~/ftf_demo# ./linux.armv8.satrace/bin/ls.linux.satrace -v ./ftfLinuxDemo.elf
User space trace
Application  : `./ftfLinuxDemo.elf`
Arguments    :
Starting `./ftfLinuxDemo.elf`
Hello World!
User application exit status : 0
Master process
Relocation file  : `/home/root/ftf_demo/ftfLinuxDemo.rlog`
Trace file  : `/home/root/ftf_demo/ftfLinuxDemo.dat`
Collecting trace ...
Archive file  : `/home/root/ftf_demo/ftfLinuxDemo.cwzsa`
Creating archive ....
Archiving /home/root/ftf_demo/ftfLinuxDemo.dat
Archiving /home/root/ftf_demo/ftfLinuxDemo.elf
Archiving /home/root/ftf_demo/ftfLinuxDemo.rlog
Archiving /home/root/ftf_demo/linux.armv8.satrace/config/PlatformConfig.xml
Archiving /home/root/ftf_demo/linux.armv8.satrace/bin/ftfLinuxDemo.resultsConfig
Archiving /lib/ld-2.19-2014.04.so
Archiving /lib/libc-2.19-2014.04.so
Archiving /lib/libdl-2.19-2014.04.so
Archiving /lib/libm-2.19-2014.04.so
Archiving /lib/libpthread-2.19-2014.04.so
Archiving /lib/librt-2.19-2014.04.so
root@ls2085aqds:~/ftf_demo# ls
ftfLinuxDemo.cwzsa   ftfLinuxDemo.elf     linux.armv8.satrace
root@ls2085aqds:~/ftf_demo#
```

Linux probeless trace *(CodeWarrior)*

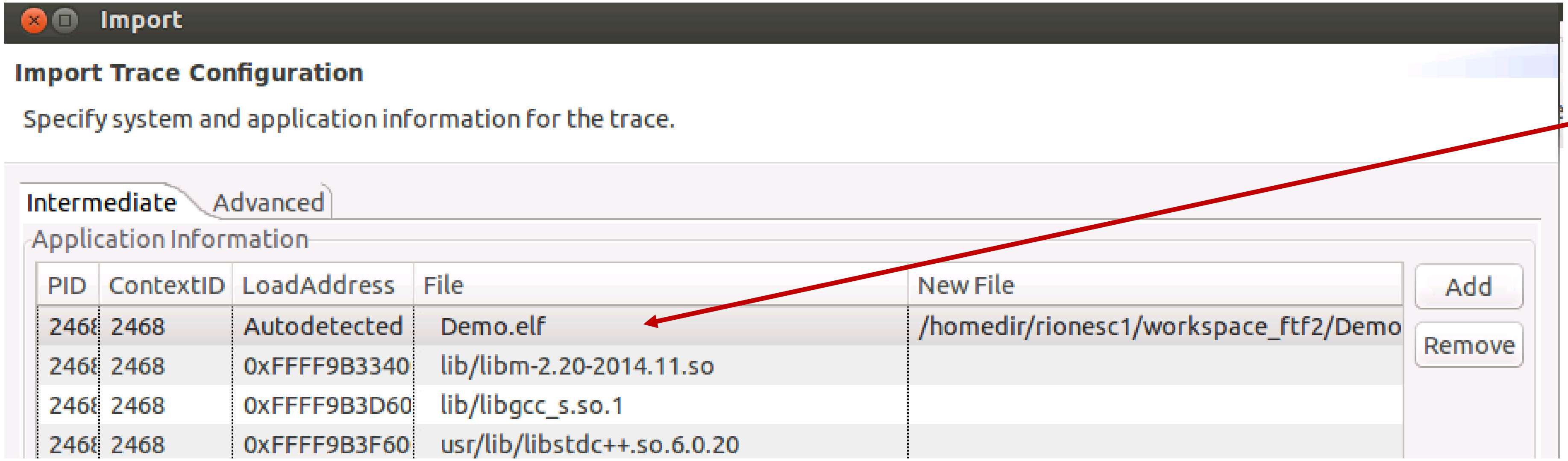
10. Refresh the files system view. Notice the newly created *.cwzsa file. Double-click on it to import trace results on host.



- ftf_demo
  - linux.armv8.satrace
  - ftfLinuxDemo.cwzsa
  - ftfLinuxDemo.elf

Trace results file

11. During import process, select the right binary file:



Import Trace Configuration

Specify system and application information for the trace.

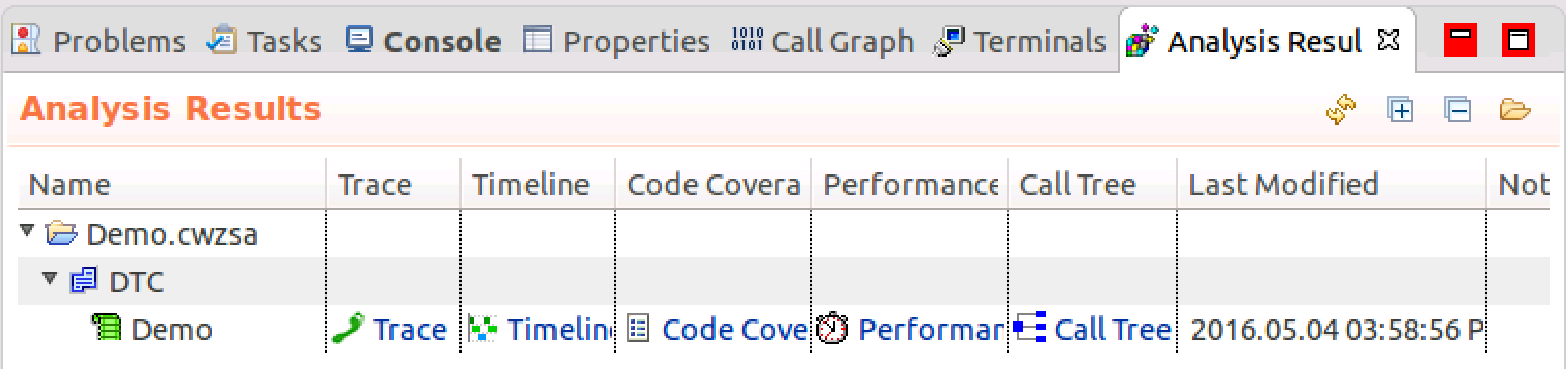
Application Information

PID	ContextID	LoadAddress	File	New File
2468	2468	Autodetected	Demo.elf	/homedir/rionesc1/workspace_ftf2/Demo
2468	2468	0xFFFF9B3340	lib/libm-2.20-2014.11.so	
2468	2468	0xFFFF9B3D60	lib/libgcc_s.so.1	
2468	2468	0xFFFF9B3F60	usr/lib/libstdc++.so.6.0.20	

Binary file

Linux probeless trace *(CodeWarrior)*

12. Notice the Analysis Results view in CodeWarrior. Browse the results and open them.



Name	Trace	Timeline	Code Covera	Performance	Call Tree	Last Modified	Not
Demo.cwzsa							
DTC							
Demo	Trace	Timelin	Code Cove	Performar	Call Tree	2016.05.04 03:58:56 P	

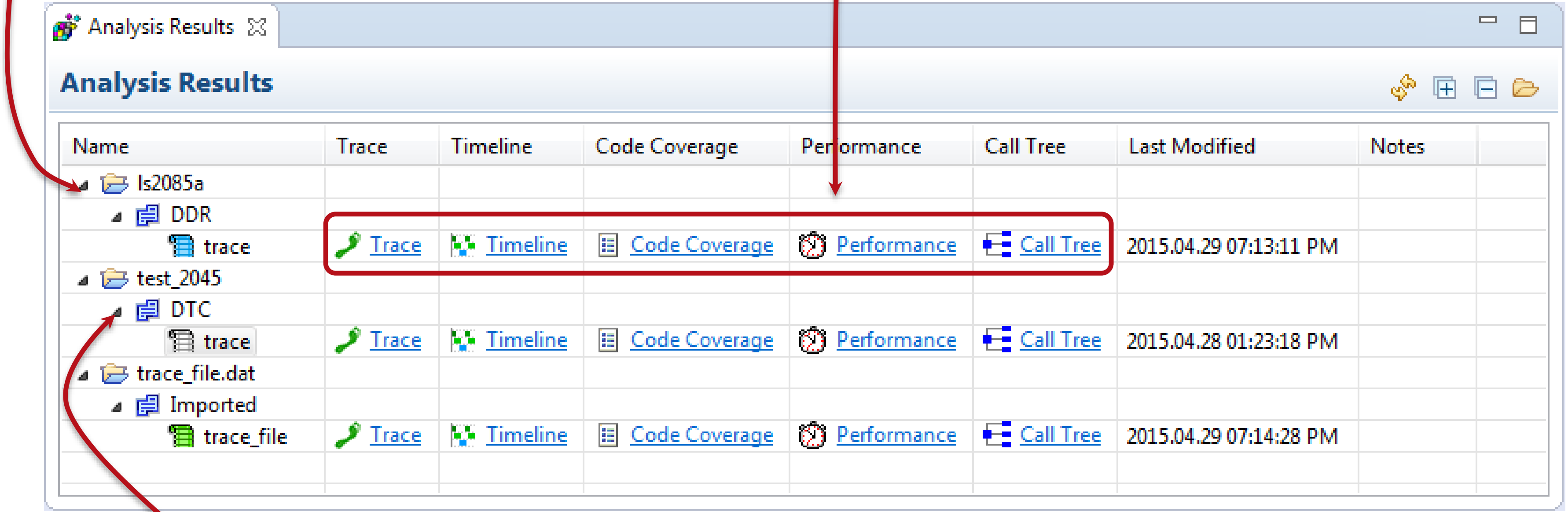
LINUX TRACE – VIEW RESULTS

Analysis Results content

Links

- ✓ Trace Viewer
- ✓ Timeline
- ✓ Code Coverage
- ✓ Hierarchical Performance
- ✓ Call Tree

Platform configuration
used to collect trace



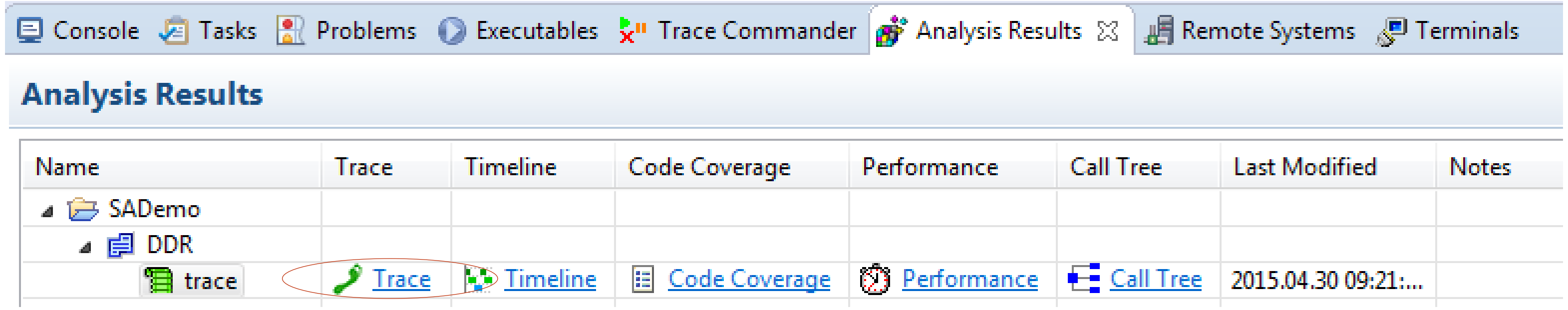
Name	Trace	Timeline	Code Coverage	Performance	Call Tree	Last Modified	Notes
ls2085a							
DDR							
trace	Trace	Timeline	Code Coverage	Performance	Call Tree	2015.04.29 07:13:11 PM	
test_2045							
DTC							
trace	Trace	Timeline	Code Coverage	Performance	Call Tree	2015.04.28 01:23:18 PM	
trace_file.dat							
Imported							
trace_file	Trace	Timeline	Code Coverage	Performance	Call Tree	2015.04.29 07:14:28 PM	

Data source from where the trace was collected:

- ➢ DTC
- ➢ DDR
- ➢ or Imported trace

Trace Viewer

1. From results folder:

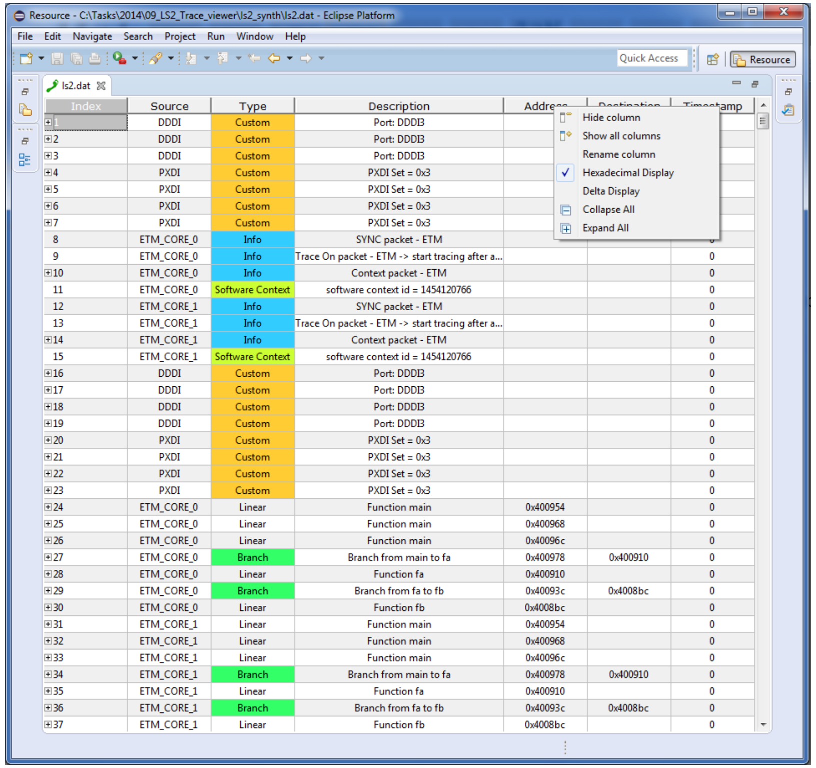


Analysis Results

Name	Trace	Timeline	Code Coverage	Performance	Call Tree	Last Modified	Notes
SADemo							
DDR							
trace	Trace	Timeline	Code Coverage	Performance	Call Tree	2015.04.30 09:21:...	

2. Open Trace Viewer:

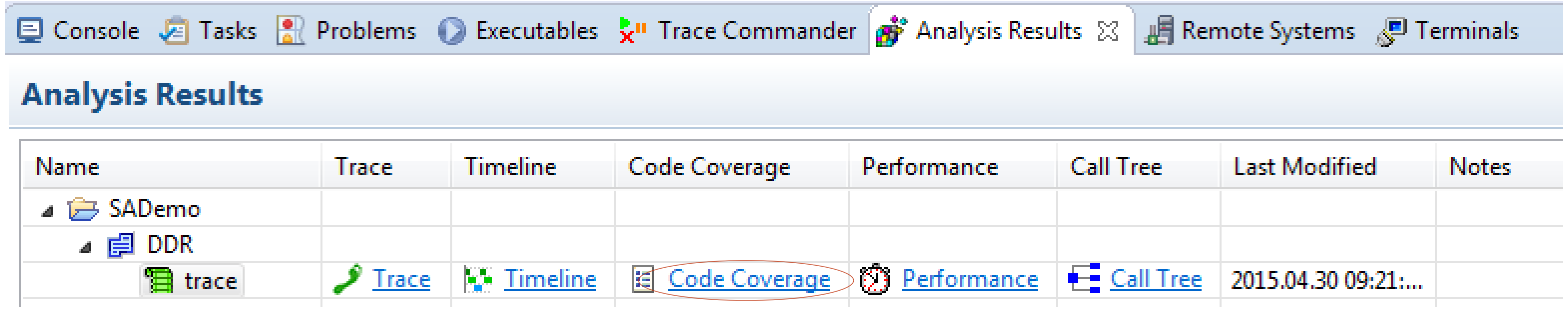
✓ Accurate information about program flow, DDR transactions, instrumentation trace, NoC transactions and PCI Express debug status.



Index	Source	Type	Description	Address	Destination	Timestamp
1	DDDI	Custom	Port: DDDI3			
2	DDDI	Custom	Port: DDDI3			
3	DDDI	Custom	Port: DDDI3			
4	PXDI	Custom	PXDI Set = 0x3			
5	PXDI	Custom	PXDI Set = 0x3			
6	PXDI	Custom	PXDI Set = 0x3			
7	PXDI	Custom	PXDI Set = 0x3			
8	ETM_CORE_0	Info	SYNC packet - ETM			0
9	ETM_CORE_0	Info	Trace On packet - ETM -> start tracing after a...			0
10	ETM_CORE_0	Info	Context packet - ETM			0
11	ETM_CORE_0	Software Context	software context id = 1454120766			0
12	ETM_CORE_1	Info	SYNC packet - ETM			0
13	ETM_CORE_1	Info	Trace On packet - ETM -> start tracing after a...			0
14	ETM_CORE_1	Info	Context packet - ETM			0
15	ETM_CORE_1	Software Context	software context id = 1454120766			0
16	DDDI	Custom	Port: DDDI3			0
17	DDDI	Custom	Port: DDDI3			0
18	DDDI	Custom	Port: DDDI3			0
19	DDDI	Custom	Port: DDDI3			0
20	PXDI	Custom	PXDI Set = 0x3			0
21	PXDI	Custom	PXDI Set = 0x3			0
22	PXDI	Custom	PXDI Set = 0x3			0
23	PXDI	Custom	PXDI Set = 0x3			0
24	ETM_CORE_0	Linear	Function main	0x400954		0
25	ETM_CORE_0	Linear	Function main	0x400968		0
26	ETM_CORE_0	Linear	Function main	0x40096c		0
27	ETM_CORE_0	Branch	Branch from main to fa	0x400978	0x400910	0
28	ETM_CORE_0	Linear	Function fa	0x400910		0
29	ETM_CORE_0	Branch	Branch from fa to fb	0x40093c	0x4008bc	0
30	ETM_CORE_0	Linear	Function fb	0x4008bc		0
31	ETM_CORE_1	Linear	Function main	0x400954		0
32	ETM_CORE_1	Linear	Function main	0x400968		0
33	ETM_CORE_1	Linear	Function main	0x40096c		0
34	ETM_CORE_1	Branch	Branch from main to fa	0x400978	0x400910	0
35	ETM_CORE_1	Linear	Function fa	0x400910		0
36	ETM_CORE_1	Branch	Branch from fa to fb	0x40093c	0x4008bc	0
37	ETM_CORE_1	Linear	Function fb	0x4008bc		0

Code coverage

1. From results folder:



Console · Tasks · Problems · Executables · Trace Commander · Analysis Results ☒ · Remote Systems · Terminals

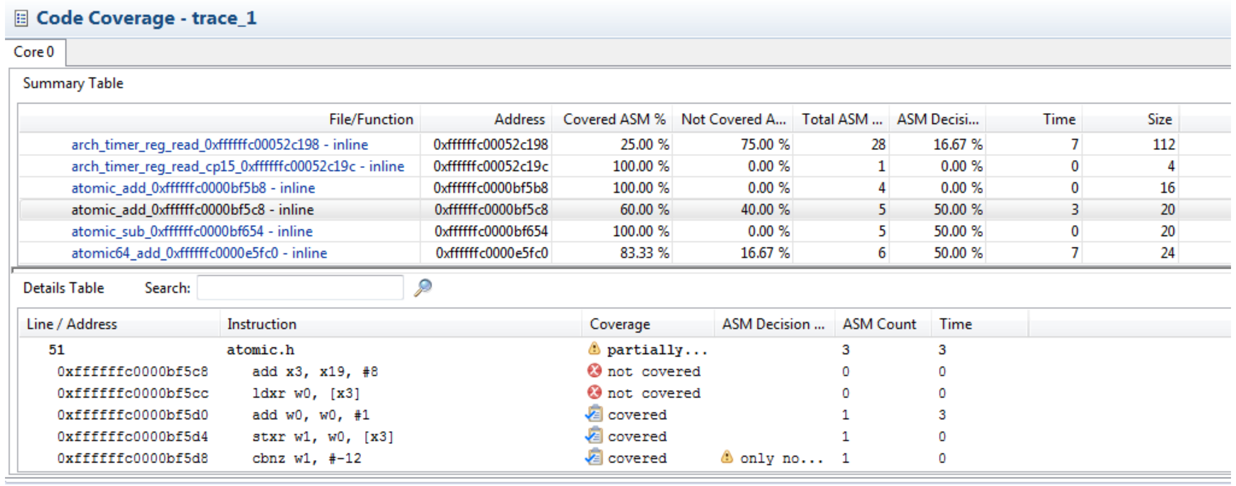
Analysis Results

Name	Trace	Timeline	Code Coverage	Performance	Call Tree	Last Modified	Notes
◢ SADemo							
◢ DDR							
trace	Trace	Timeline	Code Coverage	Performance	Call Tree	2015.04.30 09:21:...	

2. Open code coverage viewer:

Split pane with 2 types of info:

- ✓ Summary table displaying statistics for each function
- ✓ Details table displaying line-by-line coverage of selected function



Code Coverage - trace_1

Core 0

Summary Table

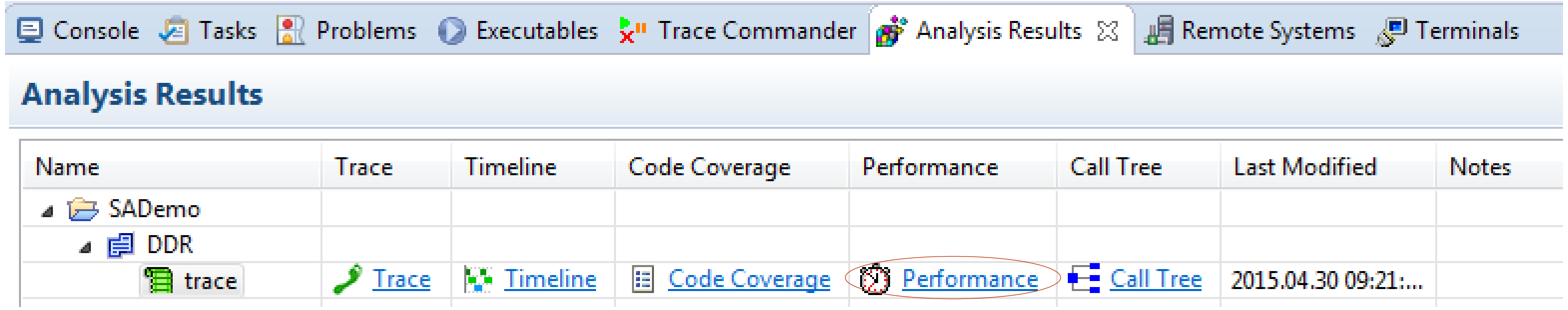
File/Function	Address	Covered ASM %	Not Covered A...	Total ASM ...	ASM Decisi...	Time	Size
arch_timer_reg_read_0xffffffc00052c198 - inline	0xffffffc00052c198	25.00 %	75.00 %	28	16.67 %	7	112
arch_timer_reg_read_cp15_0xffffffc00052c19c - inline	0xffffffc00052c19c	100.00 %	0.00 %	1	0.00 %	0	4
atomic_add_0xffffffc0000bf5b8 - inline	0xffffffc0000bf5b8	100.00 %	0.00 %	4	0.00 %	0	16
atomic_add_0xffffffc0000bf5c8 - inline	0xffffffc0000bf5c8	60.00 %	40.00 %	5	50.00 %	3	20
atomic_sub_0xffffffc0000bf654 - inline	0xffffffc0000bf654	100.00 %	0.00 %	5	50.00 %	0	20
atomic64_add_0xffffffc0000e5fc0 - inline	0xffffffc0000e5fc0	83.33 %	16.67 %	6	50.00 %	7	24

Details Table Search:

Line / Address	Instruction	Coverage	ASM Decision ...	ASM Count	Time
51	atomic.h	⚠ partially...		3	3
0xffffffc0000bf5c8	add x3, x19, #8	⊗ not covered		0	0
0xffffffc0000bf5cc	ldxr w0, [x3]	⊗ not covered		0	0
0xffffffc0000bf5d0	add w0, w0, #1	✓ covered		1	3
0xffffffc0000bf5d4	stxr w1, w0, [x3]	✓ covered		1	0
0xffffffc0000bf5d8	cbnz w1, #-12	✓ covered	⚠ only no...	1	0

Performance profiler

1. From results folder:

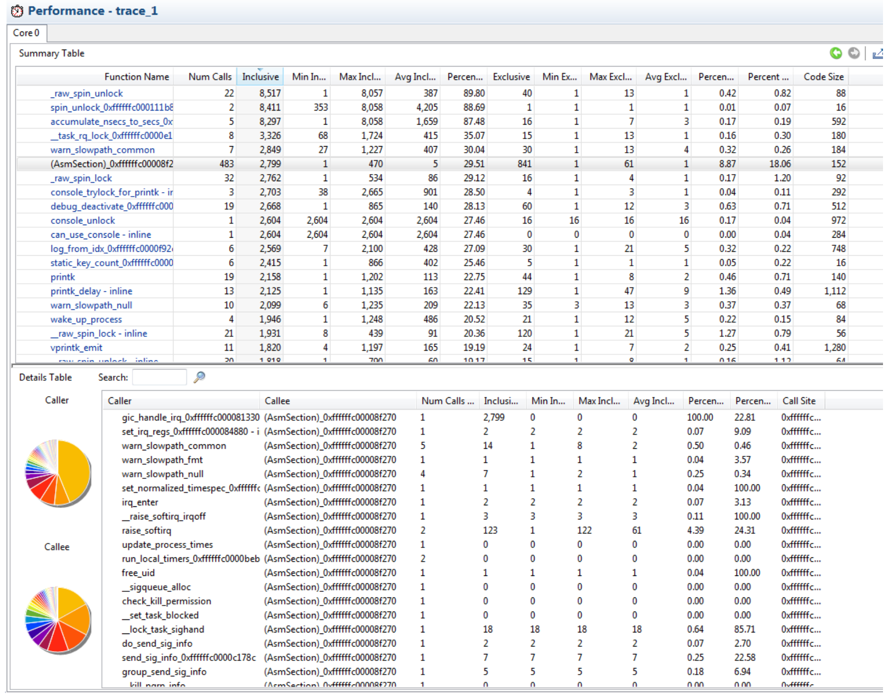


Analysis Results

Name	Trace	Timeline	Code Coverage	Performance	Call Tree	Last Modified	Notes
SADemo							
DDR							
trace	Trace	Timeline	Code Coverage	Performance	Call Tree	2015.04.30 09:21:...	

2. Open performance viewer:

- ✓ Per core analysis
- ✓ Split pane with 2 types of information:
  - Summary table displaying profiling values for functions executed in each context
  - Details table displaying performance values for caller and callee



Summary Table

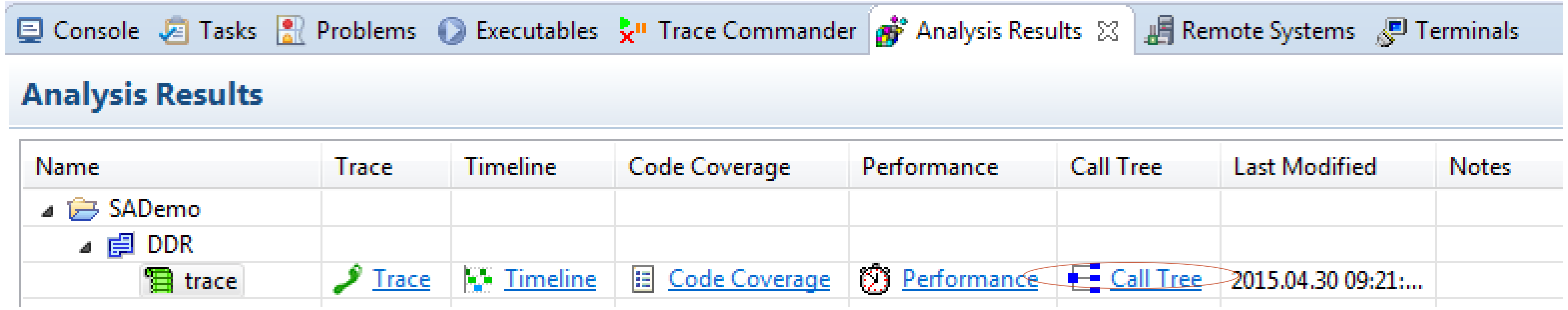
Function Name	Num Calls	Inclusive	Min In...	Max Incl...	Avg Incl...	Percen...	Exclusive	Min Ex...	Max Excl...	Avg Excl...	Percen...	Percent ...	Code Size
_raw_spin_unlock	22	8,517	1	8,057	387	89.80	40	1	13	1	0.42	0.82	88
spin_unlock_0xffffffc000111b8	2	8,411	353	8,058	4,205	88.69	1	1	1	1	0.01	0.07	16
accumulate_nsecs_to_secs_0x	5	8,297	1	8,058	1,659	87.48	16	1	7	3	0.17	0.19	592
__task_rq_lock_0xffffffc0000e1	8	3,326	68	1,724	415	35.07	15	1	13	1	0.16	0.30	180
warn_slowpath_common	7	2,849	27	1,227	407	30.04	30	1	13	4	0.32	0.26	184
(AsmSection)_0xffffffc00008f2	483	2,799	1	470	5	29.51	841	1	61	1	8.87	18.06	152
_raw_spin_lock	32	2,762	1	534	86	29.12	16	1	4	1	0.17	1.20	92
console_trylock_for_printk - ir	3	2,703	38	2,665	901	28.50	4	1	3	1	0.04	0.11	292
debug_deactivate_0xffffffc000	19	2,668	1	865	140	28.13	60	1	12	3	0.63	0.71	512
console_unlock	1	2,604	2,604	2,604	2,604	27.46	16	16	16	16	0.17	0.04	972
can_use_console - inline	1	2,604	2,604	2,604	2,604	27.46	0	0	0	0	0.00	0.04	284
log_from_idx_0xffffffc0000f92	6	2,569	7	2,100	428	27.09	30	1	21	5	0.32	0.22	748
static_key_count_0xffffffc0000	6	2,415	1	866	402	25.46	5	1	1	1	0.05	0.22	16
printk	19	2,158	1	1,202	113	22.75	44	1	8	2	0.46	0.71	140
printk_delay - inline	13	2,125	1	1,135	163	22.41	129	1	47	9	1.36	0.49	1,112
warn_slowpath_null	10	2,099	6	1,235	209	22.13	35	3	13	3	0.37	0.37	68
wake_up_process	4	1,946	1	1,248	486	20.52	21	1	12	5	0.22	0.15	84
__raw_spin_lock - inline	21	1,931	8	439	91	20.36	120	1	21	5	1.27	0.79	56
vprintk_emit	11	1,820	4	1,197	165	19.19	24	1	7	2	0.25	0.41	1,280

Details Table Search:

Caller	Callee	Num Calls ...	Inclusi...	Min In...	Max Incl...	Avg Incl...	Percen...	Percen...	Call Site
gic_handle_irq_0xffffffc000081330	(AsmSection)_0xffffffc00008f270	1	2,799	0	0	0	100.00	22.81	0xffffffc...
set_irq_regs_0xffffffc000084880 - i	(AsmSection)_0xffffffc00008f270	1	2	2	2	2	0.07	9.09	0xffffffc...
warn_slowpath_common	(AsmSection)_0xffffffc00008f270	5	14	1	8	2	0.50	0.46	0xffffffc...
warn_slowpath_fmt	(AsmSection)_0xffffffc00008f270	1	1	1	1	1	0.04	3.57	0xffffffc...
warn_slowpath_null	(AsmSection)_0xffffffc00008f270	4	7	1	2	1	0.25	0.34	0xffffffc...
set_normalized_timespec_0xffffffc	(AsmSection)_0xffffffc00008f270	1	1	1	1	1	0.04	100.00	0xffffffc...
irq_enter	(AsmSection)_0xffffffc00008f270	1	2	2	2	2	0.07	3.13	0xffffffc...
__raise_softirq_irqoff	(AsmSection)_0xffffffc00008f270	1	3	3	3	3	0.11	100.00	0xffffffc...
raise_softirq	(AsmSection)_0xffffffc00008f270	2	123	1	122	61	4.39	24.31	0xffffffc...
update_process_times	(AsmSection)_0xffffffc00008f270	1	0	0	0	0	0.00	0.00	0xffffffc...
run_local_timers_0xffffffc0000beb	(AsmSection)_0xffffffc00008f270	2	0	0	0	0	0.00	0.00	0xffffffc...
free_uid	(AsmSection)_0xffffffc00008f270	1	1	1	1	1	0.04	100.00	0xffffffc...
__sigqueue_alloc	(AsmSection)_0xffffffc00008f270	1	0	0	0	0	0.00	0.00	0xffffffc...
check_kill_permission	(AsmSection)_0xffffffc00008f270	1	0	0	0	0	0.00	0.00	0xffffffc...
__set_task_blocked	(AsmSection)_0xffffffc00008f270	1	0	0	0	0	0.00	0.00	0xffffffc...
__lock_task_sighand	(AsmSection)_0xffffffc00008f270	1	18	18	18	18	0.64	85.71	0xffffffc...
do_send_sig_info	(AsmSection)_0xffffffc00008f270	1	2	2	2	2	0.07	2.70	0xffffffc...
send_sig_info_0xffffffc0000c178c	(AsmSection)_0xffffffc00008f270	1	7	7	7	7	0.25	22.58	0xffffffc...
group_send_sig_info	(AsmSection)_0xffffffc00008f270	1	5	5	5	5	0.18	6.94	0xffffffc...

Call tree profiler

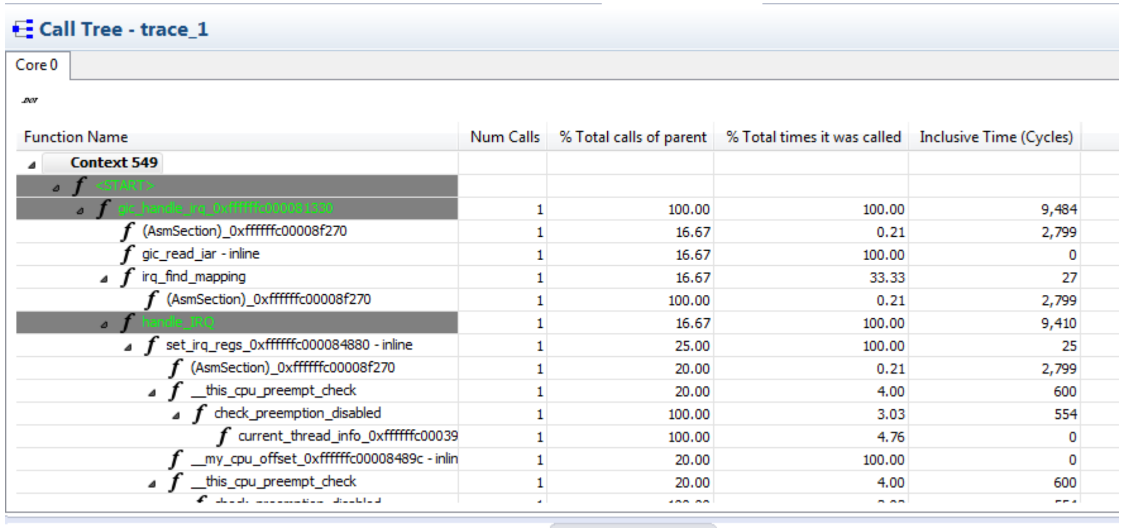
1. From results folder:



Analysis Results

Name	Trace	Timeline	Code Coverage	Performance	Call Tree	Last Modified	Notes
SADemo							
DDR							
trace	Trace	Timeline	Code Coverage	Performance	Call Tree	2015.04.30 09:21:...	

2. Open call tree viewer:



Call Tree - trace_1

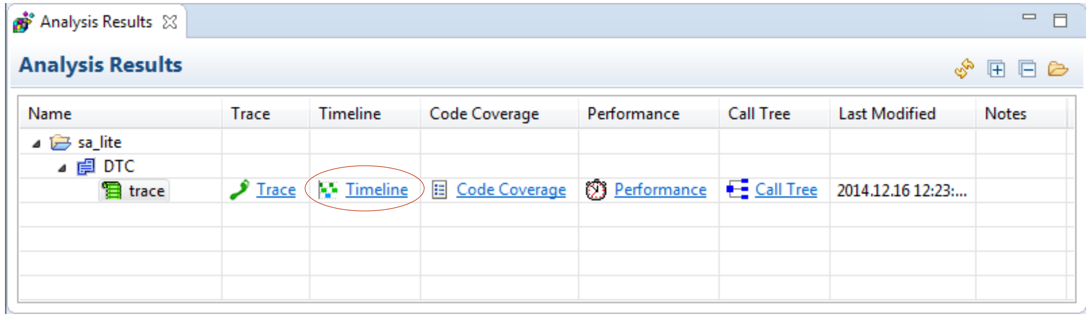
Function Name	Num Calls	% Total calls of parent	% Total times it was called	Inclusive Time (Cycles)
Context 549				
<START>				
gic_handle_irq_0xffffffc000081330	1	100.00	100.00	9,484
(AsmSection)_0xffffffc00008f270	1	16.67	0.21	2,799
gic_read_iar - inline	1	16.67	100.00	0
irq_find_mapping	1	16.67	33.33	27
(AsmSection)_0xffffffc00008f270	1	100.00	0.21	2,799
handle_IRQ	1	16.67	100.00	9,410
set_irq_regs_0xffffffc000084880 - inline	1	25.00	100.00	25
(AsmSection)_0xffffffc00008f270	1	20.00	0.21	2,799
__this_cpu_preempt_check	1	20.00	4.00	600
check_preemption_disabled	1	100.00	3.03	554
current_thread_info_0xffffffc00039	1	100.00	4.76	0
__my_cpu_offset_0xffffffc00008489c - inlin	1	20.00	100.00	0
__this_cpu_preempt_check	1	20.00	4.00	600

Shows call tree of executed functions. Two highlighted paths:

- ✓ Green color shows critical path
- ✓ Grey background shows max stack path

Timeline

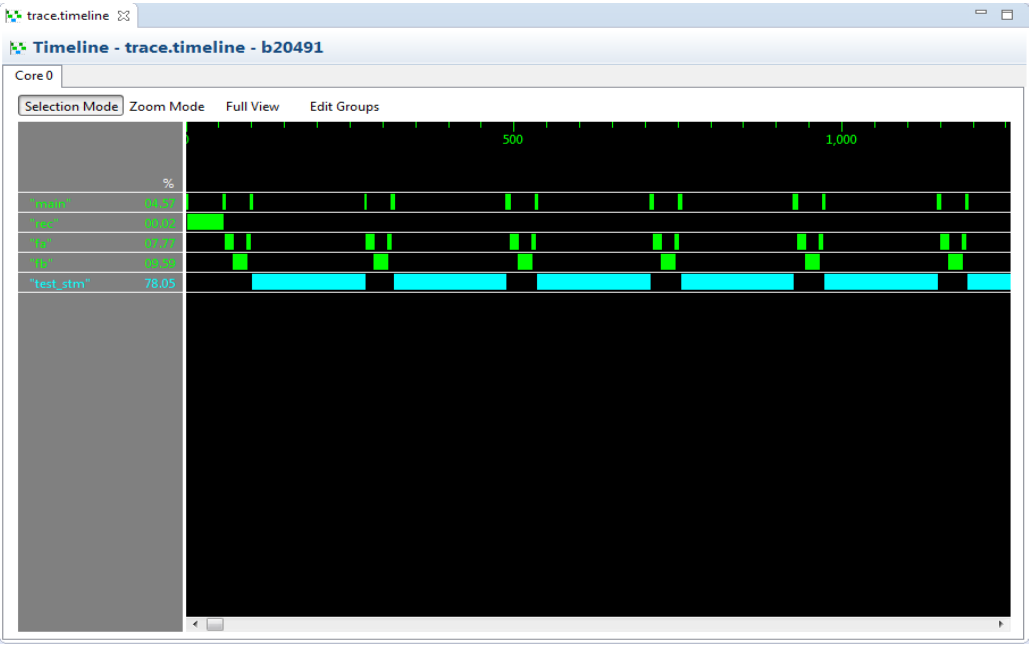
1. From Analysis Results view:



Name	Trace	Timeline	Code Coverage	Performance	Call Tree	Last Modified	Notes
sa_lite							
DTC							
trace	Trace	Timeline	Code Coverage	Performance	Call Tree	2014.12.16 12:23:...	

2. Open Timeline viewer:

- ✓ Organizes multicore results in tabs
- ✓ Customize the way the data is drawn
- ✓ Execution timeline of functions and custom groups
- ✓ Spot performance problems in code and b



SMART FILTERING

Smart filtering

There are hardware resources (e.g., comparators) available for Cortex-A57 useful to implement smart filtering. Those resources are available per core. Available resources are:

- 4 address comparator pairs
- 1 Context ID comparator
- 1 Virtual Machine ID comparator

With those resources, there are implemented 2 types of smart filters: tracepoints and ranges.

Tracepoints allow to start and stop trace collection.

Ranges(include range and exclude range) allow to enable trace collection inside or outside certain area.

Tracepoints are used when want to start trace collection at a certain address(e.g., after initialization code) or to stop trace collection after a certain address(e.g., before application printing results).

Ranges are used when want to trace always the same area, no matter how many enters or exists(e.g., a certain function).

Smart filtering

```
Usage: ./linux.armv8-sdk1.8-ear6.satrace/bin/ls.linux.satrace [Options] app [app_args]

Common options:
  -T [ --multithreading ]  Enables multithreading support.
  -p [ --pid ] PID         Attach to a process giving a PID.
  --vmid vmid              Virtual machine ID
  --start-trace address    Start tracepoint
  --stop-trace address     Stop tracepoint
  --include-range range    Include range
  --exclude-range range    Exclude range

    [range] - An interval specified using one of the following formats :
            0x2000-0x3000                Address range [0x2000, 0x3000]
            libpthread                   Executable code from libpthread.so
            init_linuxrc                 Address range based on kernel function name
                                         Covers all instructions from init_linuxrc
            init_linuxrc-init_linuxrc+8  Includes/Excludes first 8 bytes from
                                          init_linuxrc
            ipv6.ko                      Includes/Excludes 'ipv6' kernel module
    [address] - An address specified using one of the following formats :
            0x2000                       Hex address
            libpthread+200               Offset from a shared libary (libpthread.so)
            init_linuxrc                 Address based on kernel function name
            init_linuxrc+8               Kernel function offset
            ipv6.ko                      Kernel module offset
    --vmid argument is compatible only with address range filters.

Examples :
        ls.linux.satrace -p 534 --include-range=init_linuxrc-init_linuxrc+20
        ls.linux.satrace -p 534 --exclude-range=0x3000-0x7000
        ls.linux.satrace --vmid=1 -S arname -p 534 --include-range=0x2000-0x3000
                         --include-range=0x4000-0x5000
        ls.linux.satrace –K kern --start-trace=__switch_to --stop-trace=__switch_to+0x200
```

- Both tracepoints and ranges allow flexibility in defining addresses

- It can be pure addresses, libraries/executables names or symbols names

- Offsets can be used making usage more user friendly

- Also there is support for tracing kernel modules – identified by name

Smart filtering

A distinct feature is the ability to filter on VMID(Virtual Machine ID). This allow to trace only execution inside a certain virtual machine.

There is provided an experimental support for multithreading, in case of user space applications. This is related with multiple threads launched by analyzed application. There is a default mask of 256 threads that can be traced. Alternatively, only the main thread of application can be traced.

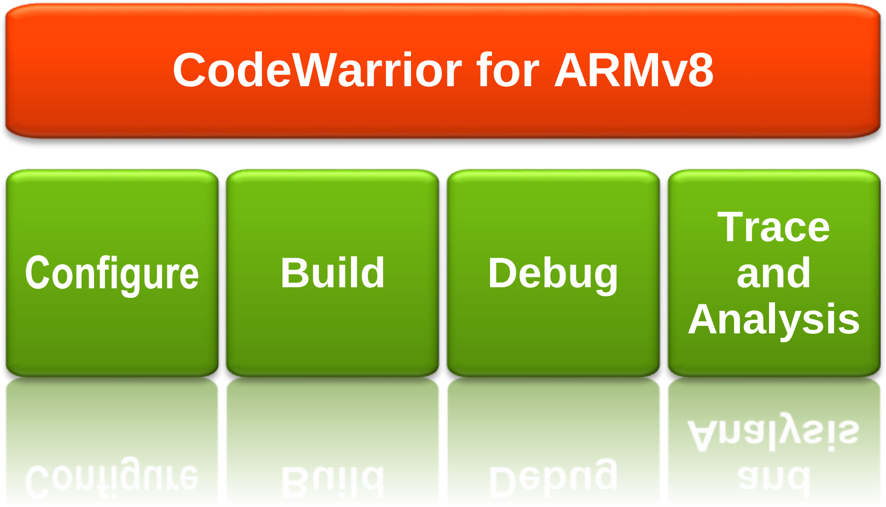
Smart filtering is a flexible mechanism to filter the amount of trace collected directly fro hardware, with no impact on application/system execution. This will greatly help on host operations(e.g., trace decoding, profile generation).

Currently available only command line, it could be very easily supported in CodeWarrior UI. The process of trace configuration is data driven and easily supports various extensions.

SUMMARY

Summary

- This course has been a brief introduction into the LS2085 RDB board and the CodeWarrior tools available to debug the board
- Linux application tracing
- Digital Networking is introducing a new networking tools suite
  - CodeWarrior Development Studio for QorIQ LS Series – ARMv8 ISA
  - Tools covering Configuration, Build, Debug, and Analysis



http://www.nxp.com/codewarrior

Q & A

SECURE CONNECTIONS
FOR A SMARTER WORLD

ATTRIBUTION STATEMENT