|  |  |
| --- | --- |
|  |  |
|  | |
| August 20, 2017. Renesas Electronics Corporation.  Specify document type and unique identifier here. (Type and identifier selection rules are described on the Wiki.)  INTEGRITY® RISP Environment Performance Evaluation Report | |

User’s Manual

[1. Revision History 4](#_Toc491775562)

[2. Project Requirements 5](#_Toc491775563)

[2.1. Software configuration 5](#_Toc491775565)

[2.2. Metrics and parameters for evaluation 6](#_Toc491775566)

[3. Virtualization PoC Implementation. Setup and Hardware Configuration 7](#_Toc491775567)

[3.1. Introduction 7](#_Toc491775568)

[3.2. Virtualization PoC Setup 7](#_Toc491775569)

[3.3. Hardware Components List 9](#_Toc491775570)

[3.4. The Salvator-X board features 10](#_Toc491775571)

[4. Virtualization PoC Implementation. Software 11](#_Toc491775572)

[4.1. Center Information 13](#_Toc491775573)

[4.1.1. 3D navigation 13](#_Toc491775574)

[4.1.2. HMI 14](#_Toc491775575)

[4.1.3. Back monitor 15](#_Toc491775576)

[4.1.4. Video/Audio playback with media player 16](#_Toc491775577)

[4.2. Instrument Cluster 17](#_Toc491775578)

[4.2.1. Meter Cluster 17](#_Toc491775579)

[4.3. Head-up display 18](#_Toc491775580)

[4.3.1. Telltale 18](#_Toc491775581)

[4.3.2. Back monitor 19](#_Toc491775582)

[5. Measurement 20](#_Toc491775583)

[5.1. CPU Load 23](#_Toc491775584)

[5.1.1. Total CPU usage on Linux 23](#_Toc491775585)

[5.1.2. Total CPU usage of Hypervisor(Multivisor) 26](#_Toc491775586)

[5.1.3. Total CPU usage on INTEGRITY 29](#_Toc491775587)

[5.1.4. The overhead (CPU usage) compared virtualized Linux with native Linux 32](#_Toc491775588)

[5.1.5. Overhead API Forwarding performance (for Hypervisor) 36](#_Toc491775589)

[5.1.6. Math operation (for Hypervisor) 39](#_Toc491775590)

[5.2. Bus Load/Bandwidth 41](#_Toc491775591)

[5.2.1. Total bus bandwidth on virtualization environment 41](#_Toc491775592)

[5.2.2. Total bus bandwidth on native Linux environment 43](#_Toc491775593)

[5.2.3. Total bus bandwidth on native INTEGRITY environment 45](#_Toc491775594)

[5.2.4. The overhead (DDR memory bandwidth) compared virtualized Linux with native Linux 46](#_Toc491775595)

[5.3. Boot Time 48](#_Toc491775638)

[5.3.1. From power on to booting of INTEGRITY OS 49](#_Toc491775639)

[5.3.2. From power on to starting up of Meter cluster application on INTEGRITY 51](#_Toc491775640)

[5.3.3. From power on to booting of Linux OS 53](#_Toc491775641)

[5.3.4. From power on to starting up of Video app and MAP/HMI of graphics on Linux OS 55](#_Toc491775642)

[5.4. Interrupt Time 57](#_Toc491775643)

[5.4.1. Delay time for interrupt 57](#_Toc491775644)

[5.4.2. Delay time variation 59](#_Toc491775645)

[5.4.3. Lock Synchronization latency 59](#_Toc491775646)

[5.5. RAM I/O Performance 60](#_Toc491776443)

[5.5.1. RAM I/O Performance 60](#_Toc491776444)

[5.6. Memory Performance 63](#_Toc491776445)

[5.6.1. Sequential reading performance 63](#_Toc491776446)

[5.6.2. Sequential writing performance 70](#_Toc491776447)

[5.6.3. Random reading performance 72](#_Toc491776448)

[5.6.4. Random writing performance 77](#_Toc491776449)

[5.6.5. Memory Allocate/Deallocate performance 81](#_Toc491776450)

[5.6.6. Read Cached/Uncached memory performance 85](#_Toc491776451)

[5.6.7. TLB(Translation look aside buffer) miss performance 86](#_Toc491776452)

[5.6.8. VA - IPA -PA conversion performance 89](#_Toc491776453)

[5.7. RTOS performance 91](#_Toc491776489)

[5.7.1. INTEGRITY OS Performance 91](#_Toc491776490)

[5.8. Memory usage 95](#_Toc491776743)

[5.8.1. Check the memory usage of Multivisor 95](#_Toc491776744)

[5.9. Memory Utilization of Each Module 101](#_Toc491776803)

[5.9.1. Memory utilization in IVI (Center Information) 101](#_Toc491776804)

[5.9.2. Memory utilization in meter (Instrument Cluster) 102](#_Toc491776805)

[5.9.3. Memory utilization in HUD (Head-up display) 106](#_Toc491776806)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| rev | Date | Description | Note |
| 1.0 | August 20, 2017 | Created first release |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Project Requirements

RISP System Integration project scope and requirements are based on the following documents:

* R-CarH3\_2ndSI\_INTEGRITY\_RISP\_Package\_T15\_0\_v0\_8\_1.pptx

## Software configuration

The following figure software configuration for evaluation.

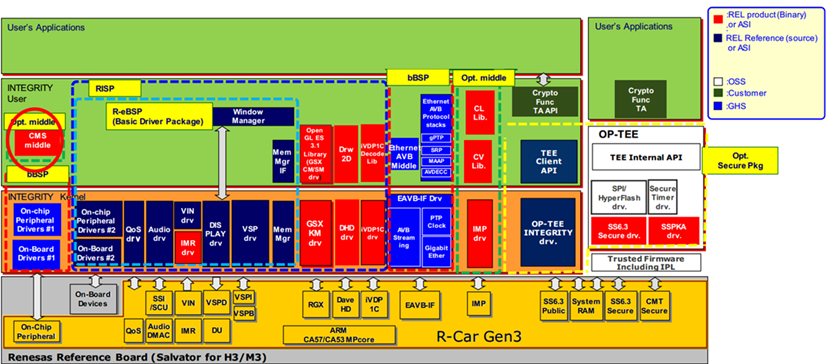


Figure 2‑1: Image of Software configuation

## (need review) Metrics and parameters for evaluation

RISP System Load and Performance evaluates the following Focus Points.

* Use case (System Integration test)
* CPU Load
* Memory Load/Bandwidth
* Boot Time
* RAM I/O Performance
* Memory Performance
* Application Switching performance
* Memory usage
* Memory Utilization of Each Module

# INTEGRITY RISP V0.8.1 Demo. Setup and Hardware Configuration

## Introduction

This chapter describes hardware required for INTEGRITY RISP V0.8.1 setup.

## INTEGRITY RISP V0.8.1 Setup

The following figure shows the INTEGRITY RISP setup.

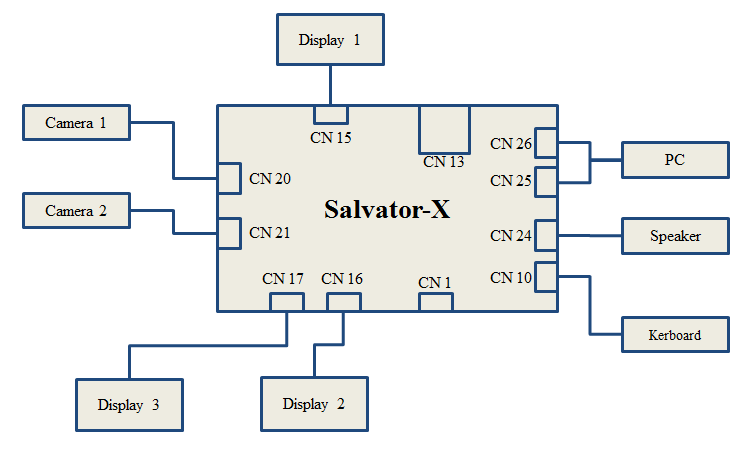


Figure 3‑1: INTEGTIY RISP setup

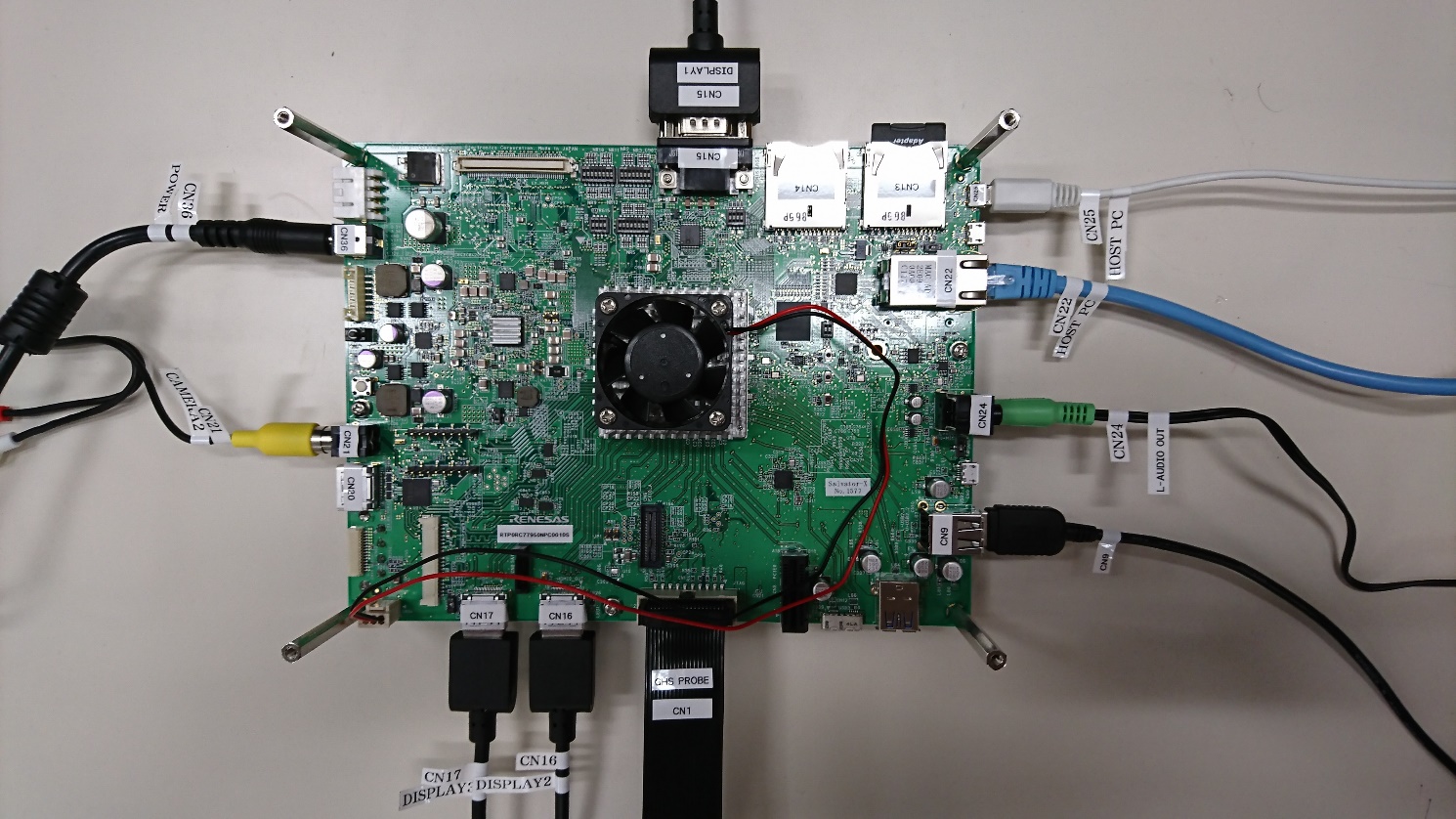


Figure 3‑2: INTEGRITY RISP top view

Table 3‑1: Connector List

|  |  |  |
| --- | --- | --- |
| **Connector** | **Item** | **Description** |
| CN 10 | Keyboard | Insert a USB-Keyboard |
| CN 13 | SD Card | Insert a SD card |
| CN 15 | Display 1 | Connect an Analog RGB Display (use for display Head-up display) |
| CN 16 | Display 2 | Connect a HDMI Display (use for display Center Information) |
| CN 17 | Display 3 | Connect a HDMI Display (use for display Instrument Cluster) |
| CN 21 | Camera 2 | Connect a NTSC Camera |
| CN 24 | Speaker | Connect a stereo speaker to AudioOut |
| CN 25/CN26 | PC | Connect a PC (use for terminal software on PC) |

## (Review) Hardware Components List

The following Table shows the virtualization Hardware components.

Table 3‑2: Hardware Components List

|  |  |  |
| --- | --- | --- |
| **Item** | **Quantity** | **Description** |
| Salvator-X board | 1 | R-Car H3-SiP System Evaluation Board |
| Display | 3 | On-Lap 1303I, 13.3 inch, microHDMI support |
| Speaker | 1 | SANWA MM-SPL2N2 |
| USB charger | 1 | SANWA 700-AC011-W |
| Power Supply TAP | 1 | SANWA TAP-TSH61N |
| SD card (\*) | 7 | IO-DATA: SDMCH-W8G/A x5,  IO-DATA: BMS-8G10RW x2 |
| Camera | 1 | Gopro HERO3+ miniHDMI,miniUSB support |
| mini HDMI to HDMI Converter cable | 1 | Sony DLC-HEU10A |
| miniUSB to NTSC Converter cable | 1 | Gopro ACMPS-301 |
|  | | |
| (\*) This product includes components of the PowerVR SDK from Imagination Technologies Limited. | | |

## R-Car Gen 3 INTEGRITY RISP V0.8.1 Demo features

The following table shows the features of R-Car Gen 3 INTEGRITY RISP V0.8.1 Demo

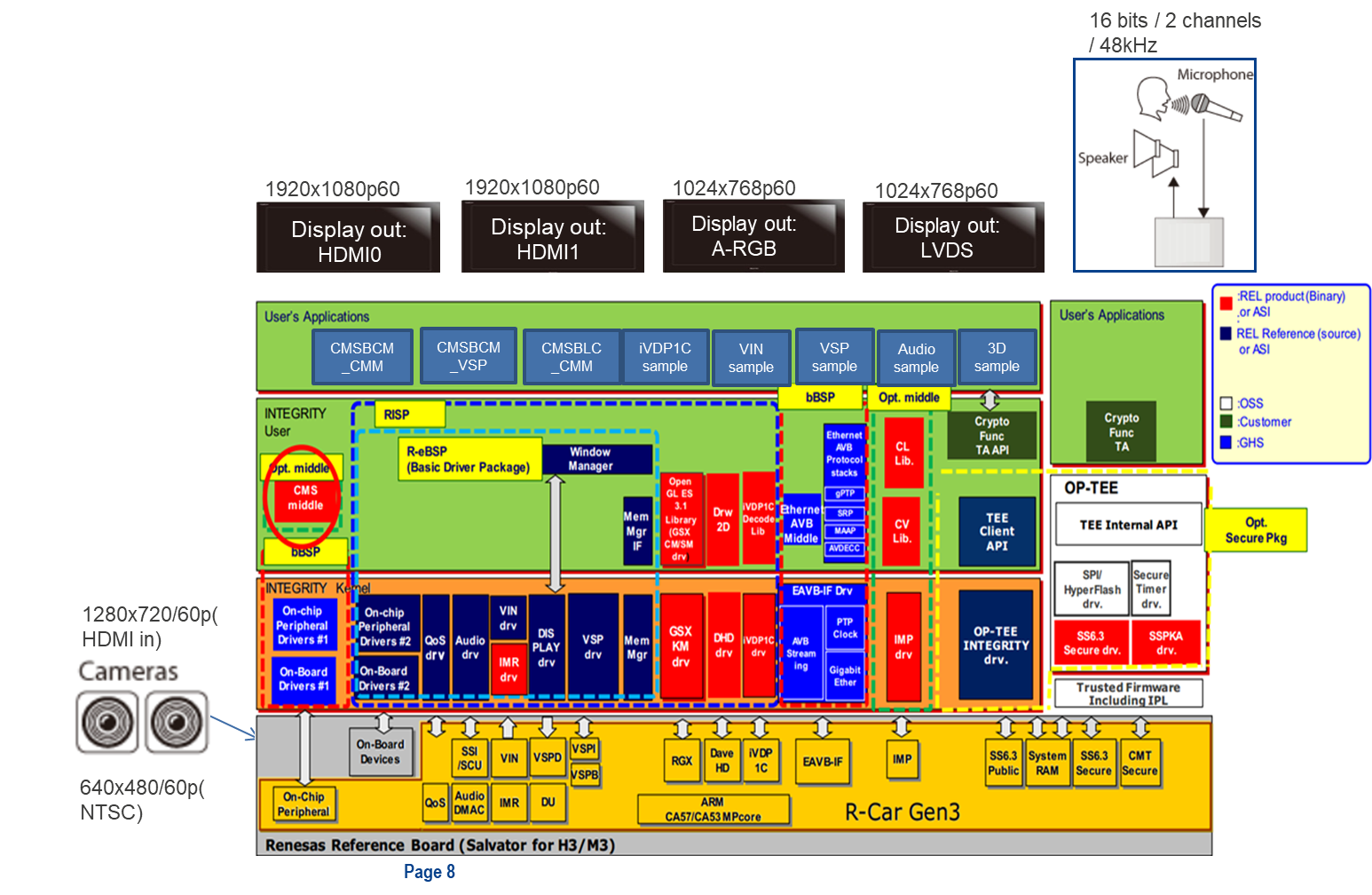
Table 3‑3: R-Car Gen 3 INTEGRITY RISP V0.8.1 Demo features

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ver | Item | Features | | INTEGRITY BSP version | Note |
| 0.8.1 | Display Unit | 3840x2160/30p(HDMI) + 1024x768/60p(LVDS) + 1024x768/60p(ARGB) simultaneously   * Display size:   VGA,SVGA,XGA,FHD,HD,4K   * Input format(VSPD input):   ARGB8888, RGBA8888, ARGB1555, RGBA5551, ARGB4444 ( Not Yet ), RGBA4444,RGB565, YUV 420, CLUT8, Alpha8.   * Output:   H3:Analog RGB(Salvator-X Board On-board DAC), HDMIx2, LVDS  M3:Analog RGB(Salvator-X Board On-board DAC), HDMIx1, LVDS   * Blend function:   Maximum 5 layers are blended for each output.   * Alpha blend:   Pixel alpha, fixed alpha, alpha blending, or 1-bit alpha converted from the color specified for pixels.  No DISCOM and DOC support | | T15.0 |  |
|  | Video Input | * 1920x1080/60p(HDMI) ( Limitation ) * 1280x720/60p(HDMI) * 640x480/60p(NTSC) * Analog via CSI1(NTSC) / HDMI via CSI0 input   Un-support 2 VINs 1280x720/60p(HDMI) + 640x480/60p(NTSC) simultaneously | |  |  |
|  | Memory manager | * Video Memory region with 32MB size * Cache functions support * No IPMMU support | |  |  |
|  | VSP module | * 1920x1080/60p(HDMI) with scale up 2 times * 5 channels ( VSPI0, VSPI1, VSPI2 , VSPBD, VSPBC ) * Support SRU/ UDS/ BRU/ HST/ HSI / LUT/ CLU ( Not yet )/ HGO/ HGT/ SHP/ RPF/ WPF | |  |  |
|  | OpenGL ES 3.1 Library | * GX6650/GX6250/GE7800/8XE 4ppc | |  |  |
|  | Support 2D GFX | * M3 only | |  |  |
|  | IMR support |  | |  |  |
|  | QoS support |  | |  |  |
|  | I2C support | * With master mode, slave mode ( not yet ) * MAX bit rate:400kbps * Target device: Video Decoder(ADV7284) + SSI CODEC(AK4643) | |  |  |
|  | Audio support | * Supports up to 10 SCU channels in independent mode * 16 bit audio samples * 2 audio channels * Uses Audio-DMAC for memory to SCU data transfer * Supports Audio-DMAC for SCU and SSI data transfer * Supports DMAPP for SCU and SSI data transfer * Asynchronous read and write support * The supported input and output sample rates are 8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000, 64000, 88200, 96000, 176400, 192000 | |  |  |
|  | iVDP1C support | * Video Decoding Processor for inter-device video transfer * Support for H.264/AVC and JPEG with 8/10/10-bit depth | |  |  |
|  | BCM support | * Support for H.264/AVC and JPEG with 8/10/10-bit depth | |  |  |

# INTEGRITY RISP V0.8.1. Software

This chapter describes software required for virtualization PoC setup.

The following figure shows the software configurations.



**Figure 4‑1: Image of Software configurations**

Virtualization PoC doesn't have Back Monitor on Head-up display. Evaluation is performed according to the specification on each OS.

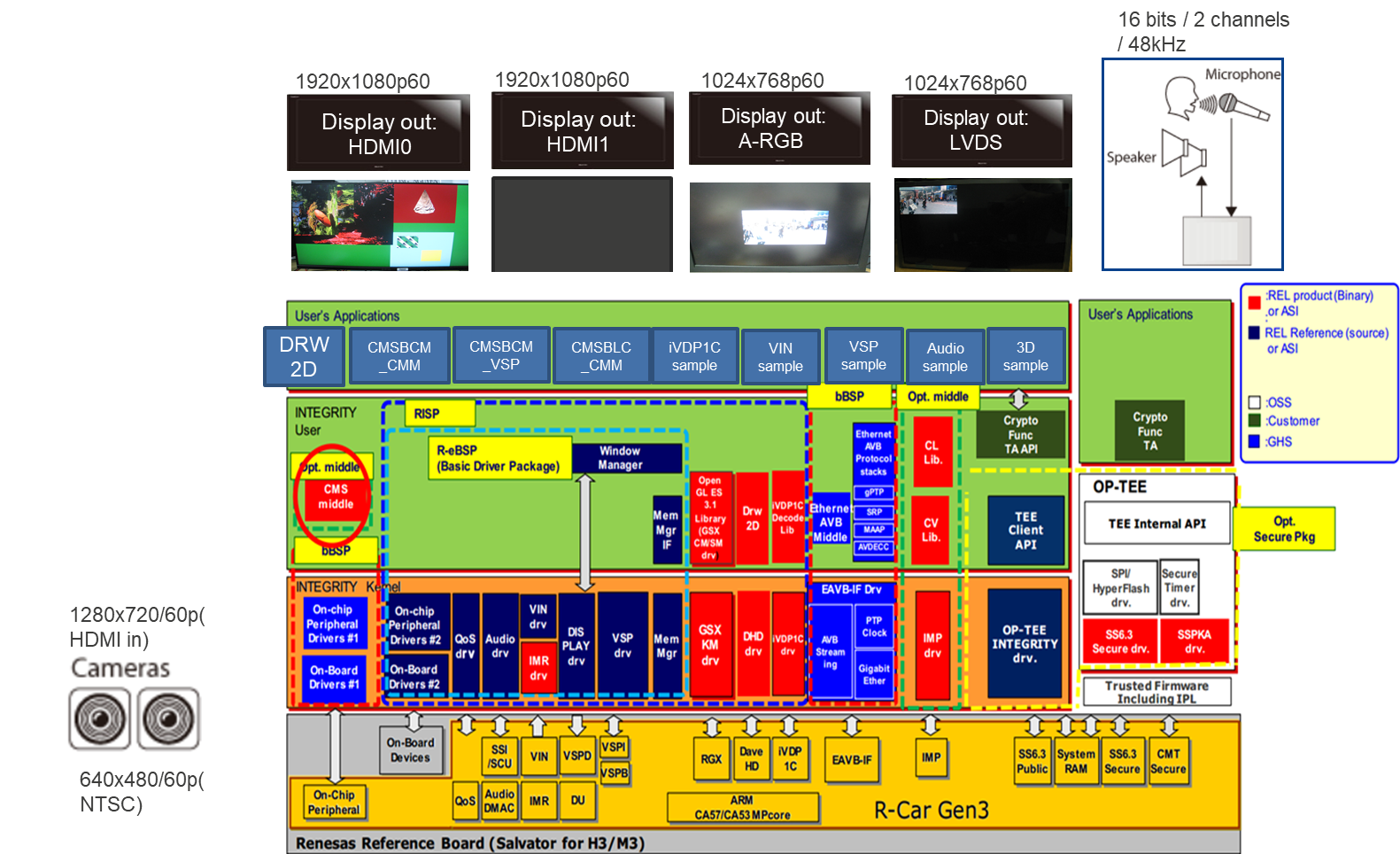


## R-Car H3 WS 2.0 – Multi app

This chapter describes software required for virtualization PoC setup.

The following figure shows the software configurations.

Figure 4‑1: Image of Software configurations



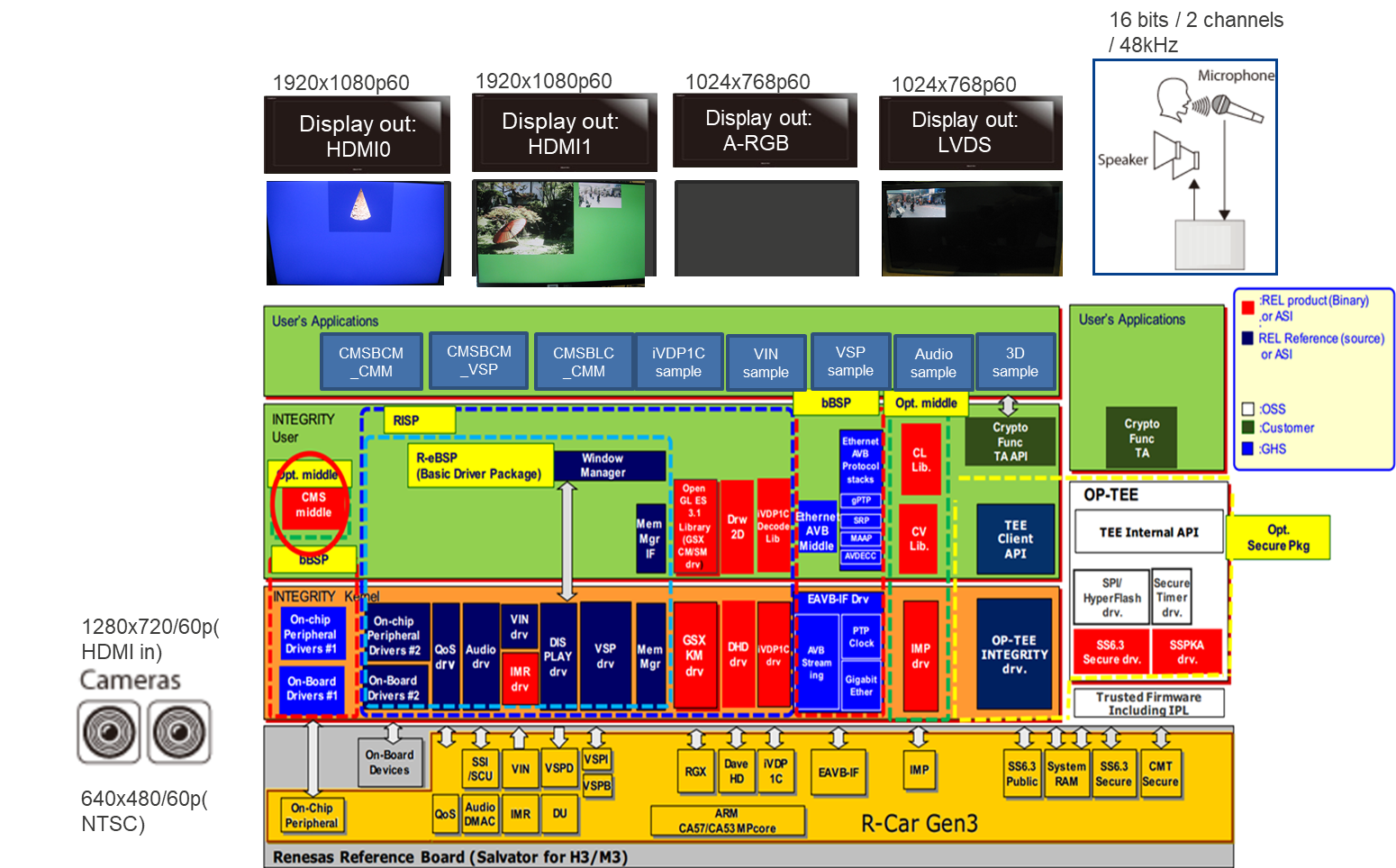
* + Stimulate sample applications run in one environment
    - [Address space 1] VSP Sample application ( HDMI 1)
    - [Address space 2 ] Audio sample application
    - [Address space 3] Es31\_sample application (HDMI0)
    - [Address space 4 ] IMR sample application
    - [Address space 5] iVDP1C sample application
    - [Address space 6] CMSBCM\_CMM application(HDMI 1)
    - [Address space 7] CMSBCM\_VSP application



## R-Car M3 WS 1.0 – Multi app

This chapter describes software required for virtualization PoC setup.

The following figure shows the software configurations.

Figure 4‑1: Image of Software configurations

* + Stimulate sample applications run in one environment
    - Address space 1] VSP Sample application ( HDMI 1)
    - [Address space 2 ] Audio sample application
    - [Address space 3] Es31\_sample application (HDMI0)
    - [Address space 4 ] IMR sample application
    - [Address space 5] iVDP1C sample application
    - [Address space 6] CMSBCM\_CMM application(HDMI 1)
    - [Address space 7] CMSBCM\_VSP application
    - [Address space 8] CMSBLC\_CMM application(LVDS)
    - [Address space 9] VIN sample (A-RGB)

## R-Car M3 WS 1.0 – Multi task

This chapter describes software required for virtualization PoC setup.

The following figure shows the software configurations.

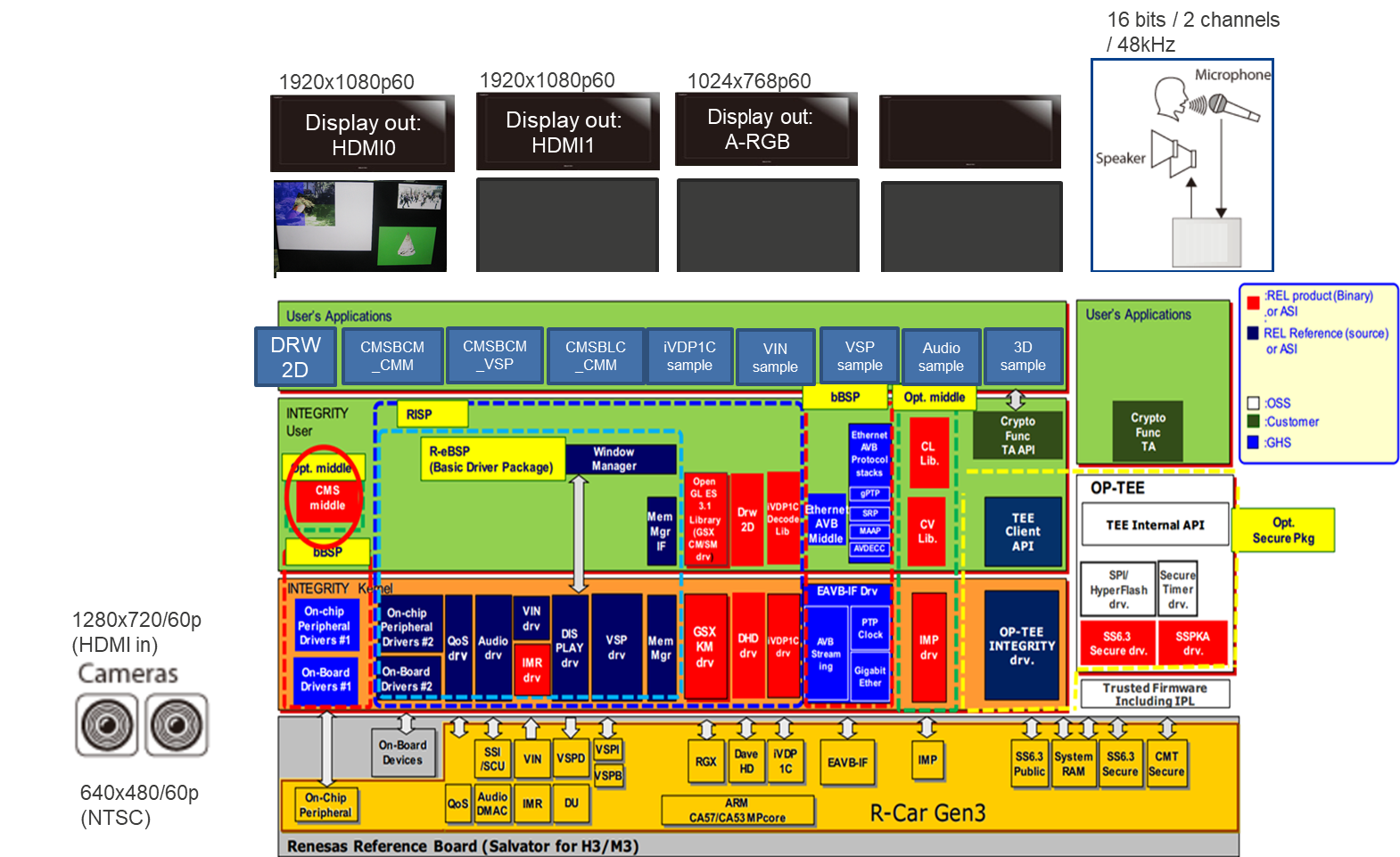


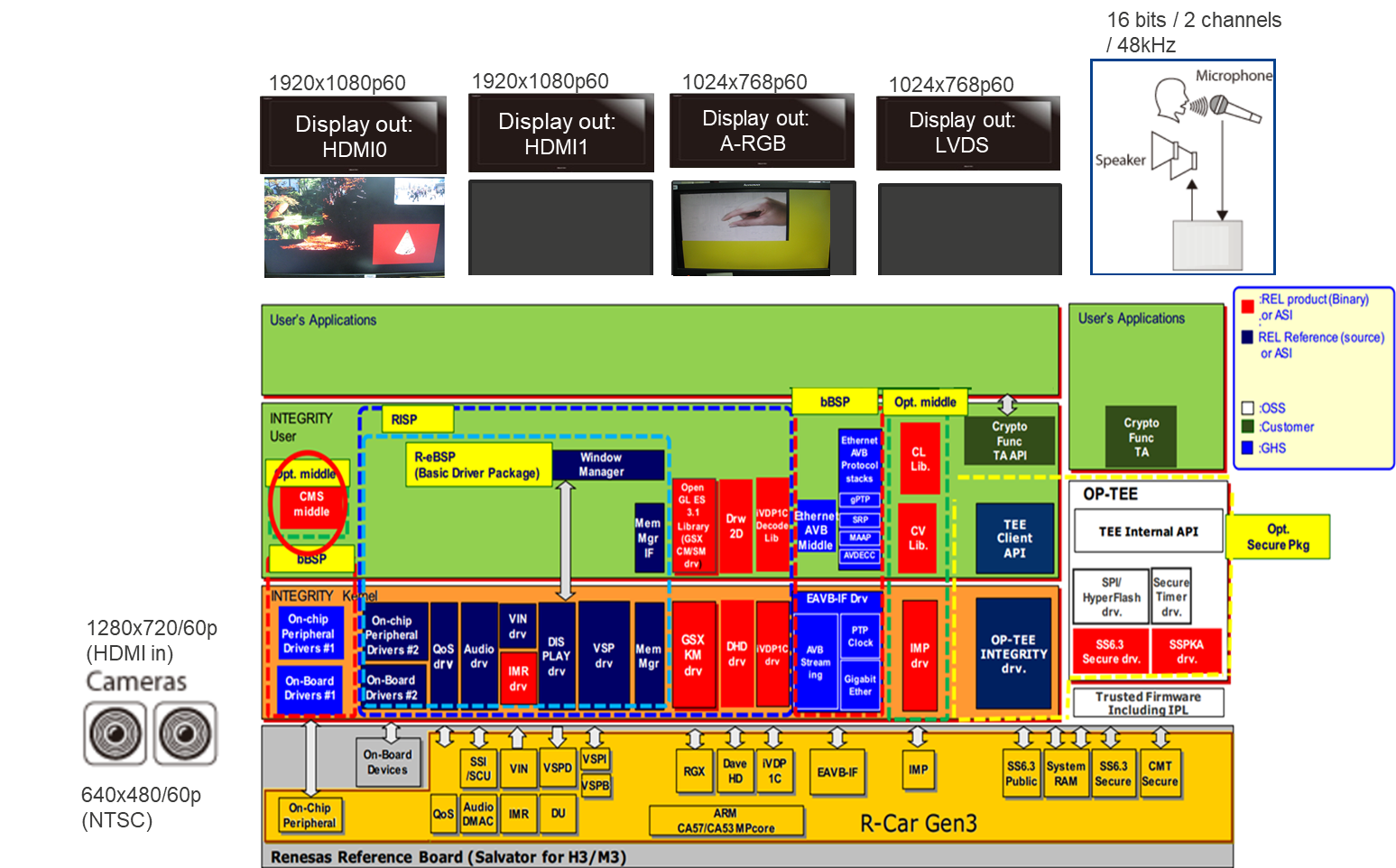
Figure 4‑1: Image of Software configurations

* Stimulate tasks in sample **application** run in one environment
* [Address space 1] 1 VSP Sample tasks + +1 CMSBCM\_CMM Sample task+ 1 DRW2D sample task + 1 3D GFX task in 1 DU screen and iVDP1C task’s log in Multi debugger
* Focus on checking Window Manager with multi-tasking, therefore, Audio task and IMR task are not added to address space 1
* Audio, IMR sample can be loaded as another apps.

## R-Car H3 WS 2.0 – Multi task

This chapter describes software required for virtualization PoC setup.

The following figure shows the software configurations.

Figure 4‑1: Image of Software configurations

* Stimulate tasks in sample **application** run in one environment
  + - [Address space 1] 1 VSP Sample tasks +1 CMSBCM\_CMM Sample task + 1 3D GFX task in 1 DU screen and 1 VIN sample task in 1 DU screen and iVDP1C task’s log in Multi debugger
      * Focus on checking Window Manager with multi-tasking, therefore, Audio task and IMR task are not added to address space 1
    - Audio, IMR sample can be loaded as another apps.























# Measurement

Executing the performance test, there are four test environments below.

If no special mentions in performance test procedure, set up a configuration that matches each type of measurement cases. Then start performance test.

INTEGRITY kernel is included debug library

*Note :*

*Type1,3,4 are using the debug library in INTEGRITY kernel in order to measure performance from INTEGRITY side. It may produce some overhead came from debug functionality.*

The following list shows the version of the measurement tool used for measurement.

Table 5‑1: measurement tool lists

|  |  |
| --- | --- |
| **Measurement Tool** | **Version** |
| lmbench | 3.0 alpha-9 |
| Multi debugging tools | MULTI 7.1.4 |
| ct command  lt command  Heapview |  |
| Api\_measurements program |  |
|  |  |

The following list shows the build configurations used for measurement.

Table 5‑2: Build configurations

|  |  |  |
| --- | --- | --- |
| **Configurations** | **Setting** | **Note** |
| INTEGRITY kernel configurations | -kernel | Use the normal kernel |
| -ldebug | Use the debug library to use some debugging functions |

## Use Case (System Integration test)

### R-Car M3 WS 1.0 INTEGRITY RISP V0.8.1

Please refer to document “INTEGRITY\_2nd\_System\_Integration\_RISP\_0\_8\_1\_TestReport.xlsx”

### R-Car H3 WS 2.0 INTEGRITY RISP V0.8.1

Please refer to document “INTEGRITY\_2nd\_System\_Integration\_RISP\_0\_8\_1\_TestReport.xlsx”

## CPU Load



























### Total CPU usage on INTEGRITY

1. Description

Measure the CPU usage of Instrument Cluster / Head-up display application on native INTEGRITY.

1. Precondition

* Measure on native INTEGRITY
* Use a tool including in Multi Debugger.

1. How to measure
2. Select [Target] – [Connect] from Menu bar
3. Select “Dynamic Download/INDRT Connection (rtserv2) for Device Tree” and press “Connect” button.
4. Select “Run mode target”
5. Run the following command on “Trg” tab.

INDRT2>ct

1. Waiting for 10 minutes.
2. Run the following command on “Trg” tab.

INDRT2>lt

After finishing a command in 6 procedures, see the log like below.

#### [INTEGRATION\_MULTITASK]

rivp\_mng 0x000000000000c000/0x0000000000023000

0xffffffa5f8075000 pending 127 0x00000000000005f0/0x0000000000002000 0.00% Initial

0xffffffa00b58a000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

0xffffffa00b581000 pending 200 0x0000000000000070/0x0000000000000dd0 0.00% Rivpd\_VdpBhTask

0xffffffa00b586000 pending 190 0x0000000000000120/0x0000000000001400 0.00% Rivp\_DepApiTask

0xffffffa00b588000 pending 190 0x0000000000000910/0x0000000000001000 0.00% Rivp\_DepJpgTask

Integration\_Multitask\_as0 0x0000000003433000/0x0000000004000000

0xffffffa5f8074000 pending 127 0x0000000000001650/0x0000000000006000 0.23% Initial

0xffffffa00b567000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

0xffffffa00b555000 exited 128 0x0000000000001220/0x000000000000fdd0 0.06% VSP\_Task

0xffffffa009349000 halted 128 0x0000000000000000/0x0000000001fffdd0 0.00% VIN

0xffffffa008dee000 pending 128 0x0000000000013b00/0x00000000004ffdd0 0.54% GFX\_Task

0xffffffa008895000 exited 128 0x0000000000000790/0x00000000004ffdd0 0.00% RIVP\_Task

DHD\_Server\_as0 0x000000000000e000/0x0000000000200000

0xffffffa5f8073000 pending 127 0x0000000000000860/0x0000000000006000 0.00% Initial

0xffffffa007567000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

0xffffffa00755c000 pending 128 0x0000000000000050/0x0000000000000dd0 0.00% DHD\_IRQ\_HANDLER

pvrserver\_as0 0x00000000000fe000/0x0000000002000000

0xffffffa5f8072000 pending 200 0x0000000000000d80/0x0000000000006000 1.61% Initial

0xffffffa007367000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

0xffffffa007345000 pending 200 0x0000000000000310/0x000000000001fdd0 0.00% pvr\_defer\_free

0xffffffa007323000 pending 200 0x0000000000000310/0x000000000001fdd0 0.00% pvr\_device\_wdg

0xffffffa007320000 pending 200 0x00000000000000d0/0x0000000000000dd0 0.03% GRAPHICS\_MISR

0xffffffa00731d000 pending 200 0x00000000000001e0/0x0000000000000dd0 0.15% GRAPHICS\_MISR

0xffffffa00731a000 pending 200 0x0000000000000260/0x0000000000000dd0 0.07% 3DGIntrTask

0xffffffa007313000 pending 200 0x00000000000001f0/0x0000000000000dd0 0.10% DC\_OS\_Task

0xffffffa007310000 pending 200 0x0000000000000100/0x0000000000000dd0 0.13% GRAPHICS\_MISR

0xffffffa00730b000 pending 200 0x0000000000000800/0x0000000000000dd0 0.09% NCGSYS\_VSYNC\_ISR

WM\_Server\_as0 0x0000000000015000/0x0000000002000000

0xffffffa5f8071000 pending 127 0x00000000000009b0/0x0000000000006000 0.61% Initial

0xffffffa005367000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

0xffffffa005364000 pending 205 0x0000000000000090/0x0000000000000dd0 0.02% DU1\_USR\_IRQ\_HANDLER

0xffffffa00535f000 pending 205 0x0000000000000040/0x0000000000000dd0 0.00% DU1\_DOCdoc\_USR\_IRQ\_HANDLER

0xffffffa00535c000 pending 205 0x0000000000000040/0x0000000000000dd0 0.00% DU1\_DOCact\_USR\_IRQ\_HANDLER

0xffffffa005359000 pending 205 0x0000000000000150/0x0000000000000dd0 0.03% VSP1\_USR\_IRQ\_HANDLER

0xffffffa005355000 pending 205 0x0000000000000090/0x0000000000000dd0 0.00% VSP1\_CMPI0\_USR\_IRQ\_HANDLER

MMGR\_Server\_as0 0x000000000004a000/0x0000000002000000

0xffffffa5f8070000 pending 127 0x00000000000005a0/0x0000000000020000 0.02% Initial

0xffffffa003367000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

This log (%) is based on one CPU base. If the all CPUs are fully loaded, the value will be 400%.

1. Result

Result is based on full CPU base. If the all CPUs are fully loaded, the value will be 100%.

Table 5‑6: Result(%)

|  |  |
| --- | --- |
| **Test environment** | **Total CPU Usage.** |
| **Native INTEGRITY (Type3)** | 0.94 |

|  |  |  |
| --- | --- | --- |
| **Test environment** | **Application** | **CPU Usage.** |
| **Native INTEGRITY (Type3)** | rivp\_mng | 0 |
| Integration\_Multitask\_as0 | 0.21 |
| DHD\_Server\_as0 | 0 |
| pvrserver\_as0 | 0.55 |
| WM\_Server\_as0 | 0.17 |
| MMGR\_Server\_as0 | 0.005 |

#### [INTEGRATION\_MULTITAS]

MMGR\_Server\_as0 0x0000000000053000/0x0000000002000000

0xffffffa5f81f2000 pending 127 0x00000000000004a0/0x0000000000006000 0.03% Initial

0xffffffa5fa1f1000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

WM\_Server\_as0 0x000000000004d000/0x0000000002000000

0xffffffa5f80b4000 pending 127 0x00000000000009b0/0x0000000000006000 5.03% Initial

0xffffffa00d07f000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

0xffffffa00d07c000 pending 205 0x00000000000000c0/0x0000000000000dd0 0.06% DU1\_USR\_IRQ\_HANDLER

0xffffffa00d077000 pending 205 0x0000000000000040/0x0000000000000dd0 0.00% DU1\_DOCdoc\_USR\_IRQ\_HANDLER

0xffffffa00d074000 pending 205 0x0000000000000040/0x0000000000000dd0 0.00% DU1\_DOCact\_USR\_IRQ\_HANDLER

0xffffffa00d071000 pending 205 0x00000000000002a0/0x0000000000000dd0 0.05% VSP1\_USR\_IRQ\_HANDLER

0xffffffa00d06d000 pending 205 0x0000000000000090/0x0000000000000dd0 0.00% VSP1\_CMPI0\_USR\_IRQ\_HANDLER

DHD\_Server\_as0 0x0000000000013000/0x0000000000040000

0xffffffa5f80b3000 pending 127 0x0000000000000940/0x0000000000002000 0.01% Initial

0xffffffa00b07f000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

0xffffffa00b074000 pending 128 0x0000000000000110/0x0000000000000dd0 0.18% DHD\_IRQ\_HANDLER

0xffffffa00b06f000 pending 128 0x0000000000000340/0x0000000000000dd0 0.38% DHD\_Client00

DRW2Dtest 0x0000000000016000/0x0000000000200000

0xffffffa5f80b2000 running 127 0x0000000000001310/0x0000000000010000 2.05% Initial

0xffffffa00b03f000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

rivp\_mng\_app 0x000000000001b000/0x0000000000440000

0xffffffa5f80b1000 exited 127 0x00000000000009c0/0x0000000000002000 0.00% Initial

0xffffffa00ae3d000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

0xffffffa00ae3a000 pending 200 0x0000000000000070/0x0000000000000dd0 0.00% Rivpd\_VdpBhTask

OES31\_Texture\_online\_Dyn\_as0 0x000000000000f000/0x0000000000500000

0xffffffa5f80b0000 pending 127 0x0000000000013b90/0x0000000000020000 1.20% Initial

0xffffffa00a9ff000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

VSP\_sample\_dyn\_as0 0x0000000000012000/0x0000000000040000

0xffffffa5f80af000 exited 127 0x0000000000001200/0x0000000000008000 0.08% Initial

0xffffffa00a4fc000 pending 128 0x0000000000000168/0x0000000000003cb8 0.00% PosixServer

0xffffffa00a4f7000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

imrlxsample\_dyn\_as0 0x0000000000003000/0x0000000000040000

0xffffffa5f80ae000 exited 127 0x00000000000007c0/0x0000000000002000 0.00% Initial

0xffffffa00a4bf000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

Audio\_sample\_dyn\_as0 0x000000000000d000/0x0000000000080000

0xffffffa5f80ad000 exited 127 0x00000000000008d0/0x0000000000002000 0.00% Initial

0xffffffa00a47f000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

CMSBLC\_CMM\_app 0x0000000000005000/0x0000000004000000

0xffffffa5f80ac000 exited 127 0x0000000000001750/0x0000000000006000 0.11% Initial

0xffffffa00a3ff000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

CMSBCM\_VSP\_app 0x000000000000b000/0x0000000000040000

0xffffffa5f80ab000 exited 127 0x00000000000010d0/0x0000000000008000 0.00% Initial

0xffffffa0063fc000 exited 128 0x0000000000000168/0x0000000000003cb8 0.00% PosixServer

0xffffffa0063f7000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

CMSBCM\_CMM\_app 0x0000000000003000/0x0000000003000000

0xffffffa5f80aa000 exited 127 0x0000000000001600/0x0000000000006000 0.00% Initial

0xffffffa0063bf000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

pvrserver\_as0 0x00000000000fe000/0x0000000002000000

0xffffffa5f80a9000 pending 200 0x0000000000000dd0/0x0000000000006000 3.15% Initial

0xffffffa0033bf000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

0xffffffa00339d000 pending 200 0x0000000000000310/0x000000000001fdd0 0.00% pvr\_defer\_free

0xffffffa00337b000 pending 200 0x0000000000000310/0x000000000001fdd0 0.00% pvr\_device\_wdg

0xffffffa003378000 pending 200 0x0000000000000330/0x0000000000000dd0 0.08% GRAPHICS\_MISR

0xffffffa003375000 pending 200 0x00000000000001e0/0x0000000000000dd0 0.28% GRAPHICS\_MISR

0xffffffa003372000 pending 200 0x00000000000002c0/0x0000000000000dd0 0.14% 3DGIntrTask

0xffffffa00336b000 pending 200 0x00000000000001f0/0x0000000000000dd0 0.26% DC\_OS\_Task

0xffffffa003368000 pending 200 0x0000000000000100/0x0000000000000dd0 0.26% GRAPHICS\_MISR

0xffffffa003363000 pending 200 0x0000000000000800/0x0000000000000dd0 0.19% NCGSYS\_VSYNC\_ISR

This log (%) is based on one CPU base. If the all CPUs are fully loaded, the value will be 400%.

1. Result

Result is based on full CPU base. If the all CPUs are fully loaded, the value will be 100%.

Table 5‑7: Result(%)

|  |  |
| --- | --- |
| **Test environment** | **Total CPU Usage.** |
| **Native INTEGRITY (Type3)** | 3.66 |

|  |  |  |
| --- | --- | --- |
| **Test environment** | **Application** | **CPU Usage.** |
| **Native INTEGRITY (Type3)** | MMGR\_Server\_as0 | 0.0075 |
| WM\_Server\_as0 | 1.29 |
| DHD\_Server\_as0 | 0.14 |
| DRW2Dtest | 0.5 |
| rivp\_mng\_app | 0 |
| OES31\_Texture\_online\_Dyn\_as0 | 0.4 |
| VSP\_sample\_dyn\_as0 | 0.2 |
| imrlxsample\_dyn\_as0 | 0 |
| Audio\_sample\_dyn\_as0 | 0 |
| CMSBLC\_CMM\_app | 0.03 |
| CMSBCM\_VSP\_app | 0 |
| CMSBCM\_CMM\_app | 0 |
| pvrserver\_as0 | 1.09 |

1. Consideration

Telltale application (DISCOM\_sample\_virt) draws periodically (3 seconds/1 frame draw), so it may result 0

to 0.5 percent depends on the measurement timing.

Meter cluster application (Sakura) uses OpenGLES library and application, and it is designed to consume

under 12.5% of CPU.



















































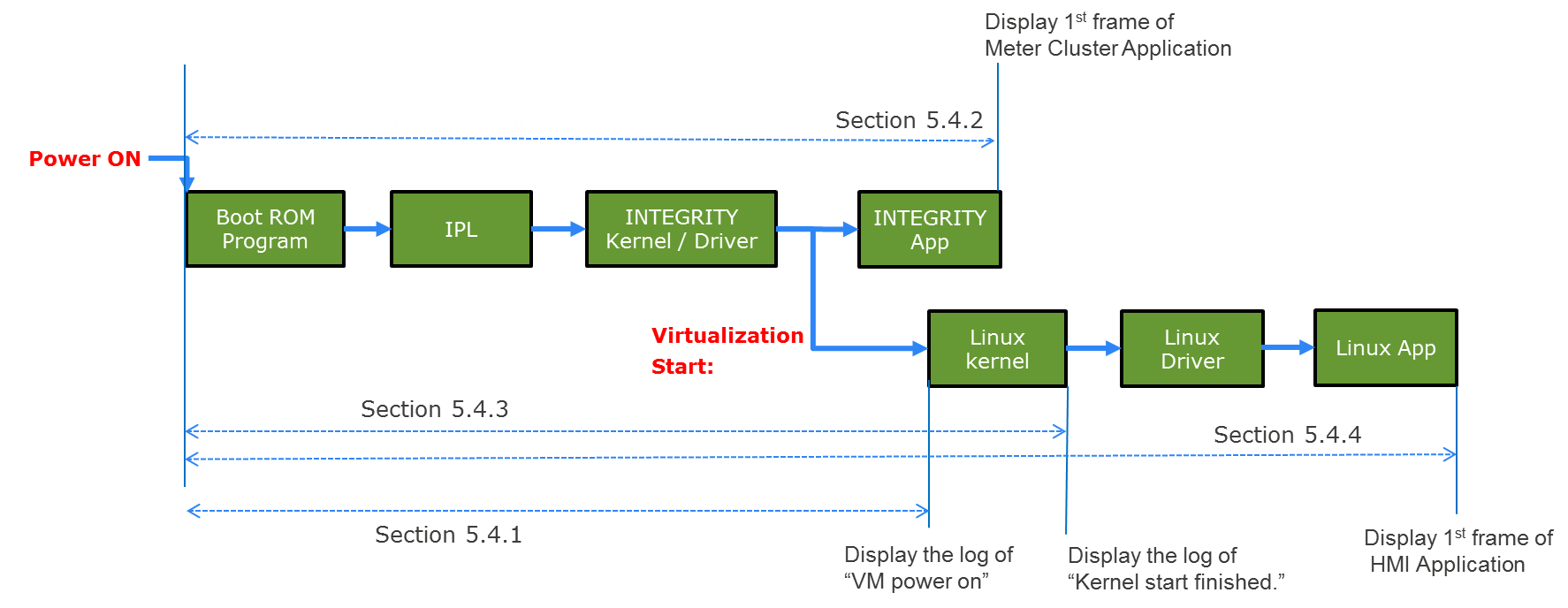




## Boot Time

This section measure the time from power on to each OS / application starts up.

The following figure describes the Booting sequence of virtualization PoC and measurement point.



|  |  |
| --- | --- |
| Legend | |
| Boot ROM Program: | Built-in mask ROM program to load the IPL from external serial flash memory. |
| IPL: | Initial Program Loader to load OS and/or U-boot from external serial flash memory. |

Figure 5‑6: Simplified BOOT Sequence

In this section, we are using following optimization to create the INTEGRITY monolith image.

- Decreasing the size of a monolith's binary in rcar\_kernel\_wrapper.readme.txt

- Multivisor Fast Boot in release\_notes.txt

This system is setting unused ram size to 112MB

ghs,size-megabytes = <112>;

Above files are provided by GHS.

Note that the virtualized Linux boot time can be reduced by using the updated INTEGRITY environment. This result is currently measured in T9.0 environment. It is known that the T10.1 release reduces boot time by approximately 1 second, and further releases are expected to reduce boot time even further.

*Attention:*

*If you have modified the HyperFlash contents of supplied Salvator-X board,*

*please write Monolith(Type1\_mono.5.4) to HyperFlash in Appendix A.*

### From power on to booting of INTEGRITY OS

1. Description

Measure the time from power on until ready of INTEGRITY Kernel to start virtual machine on virtualization PoC.

Measurement tool is boot time measurement tool. (RENESAS original)

The following figure describes measurement target.

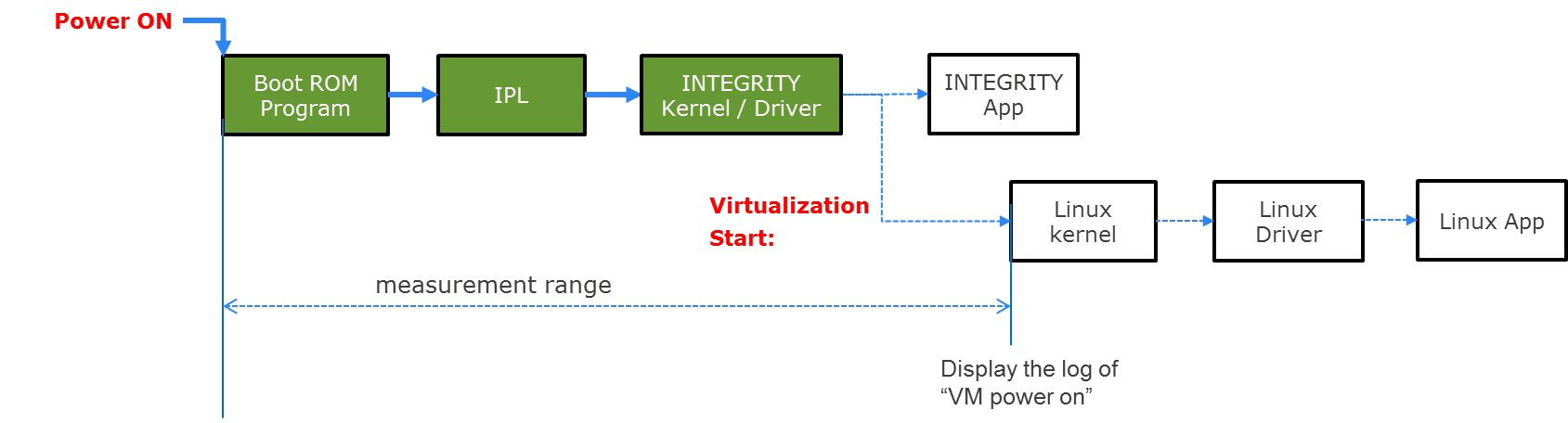


Figure 5‑7: Images of power on to booting of INTEGRITY OS

1. Precondition

* Measure on virtualization PoC (Type1)
* Verified 10 times and use the average as the result value.
* Measure time until the log of “VM power on” in INTEGRITY side.
* Take the boot situation on video.

1. How to measure
2. Take the boot situation on video.
3. Search the frame which Salvator-X LED14 changes to ON from the video.
4. Search the frame which contains the INTEGRITY Kernel log “VM powered on” from the video.

[ 2.04387] VM powered on

1. Calculate the interval time between the frames specified at the above step 2 and step 3.
2. Result

Table 5‑20: Result

|  |  |
| --- | --- |
|  | **Virtualization PoC (Type1) [sec]** |
| **Ave.** | 3.38 |
| **1** | 3.37 |
| **2** | 3.40 |
| **3** | 3.37 |
| **4** | 3.37 |
| **5** | 3.34 |
| **6** | 3.40 |
| **7** | 3.40 |
| **8** | 3.40 |
| **9** | 3.37 |
| **10** | 3.37 |

1. Consideration

In the boot time analysis, we use IPL to load the INTEGRITY OS image from HyperFlash directly, instead of loading the u-boot and use it for loading INTEGRITY OS image from either network or flash storage (SD Card, USB flash, or eMMC) to simulate the actual target hardware.

At the result of 3.38sec and the message “VM powered on” is 2.04sec so it is calculated that 1.34sec is the time before stating INTEGRITY Kernel including IPL initialization and image copy.

This time we are using the serial output on IPL for development, but actually it is possible to reduce the boot time by disabling IPL's serial message output.

The estimated boot time from power on to launch the INTEGRITY kernel is below:

INTEGRITY kernel launch time =

17ms (power-on reset time of Salvator-X board)

+ 77ms (IPL common processing time)

+ 386ms (IPL loading and validation time for 32MB INTEGRITY image from HyperFlash)

= 480ms

### From power on to starting up of Meter cluster application on INTEGRITY

1. Description

Measure the time from power on until the completion of the 1st frame.

Measurement tool is boot time measurement tool. (RENESAS original)

The following figure describes measurement target.

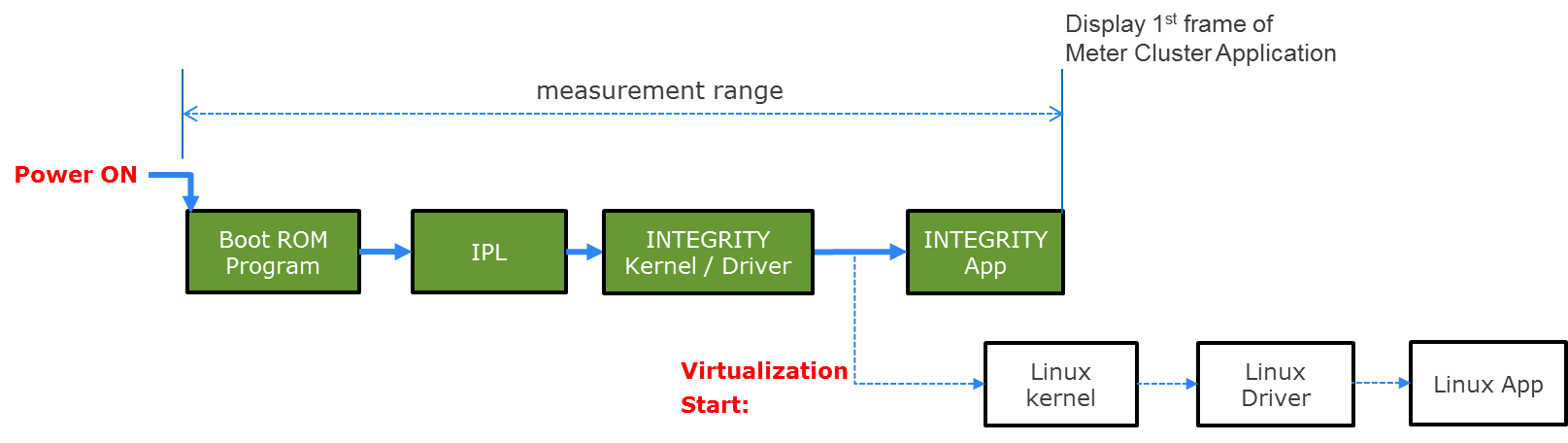


Figure 5‑8: Images of power on to starting up of Meter cluster application on INTEGRITY

1. Precondition

* Measure on virtualization PoC (Type1)
* Verified 10 times and use the average as the result value.
* Measure time until the Meter Cluster Application is launched in HDMI1 Display.
* Use the video taken at section 5.4.1.

1. How to measure
2. Take the boot situation on video.
3. Search the frame which Salvator-X LED14 changes to ON from the video.
4. Search the frame which contains a title ”R-Car” in orange color, which is output in the HDMI1 display right after the Meter Cluster Application is launched.
5. Calculate the interval time between the frames specified at the above step 1 and step 2.
6. Result

Table 5‑21: Result

|  |  |
| --- | --- |
|  | **Virtualization PoC (Type1) [sec]** |
| **Ave.** | 4.04 |
| **1** | 4.04 |
| **2** | 4.07 |
| **3** | 4.00 |
| **4** | 4.04 |
| **5** | 4.00 |
| **6** | 4.07 |
| **7** | 4.04 |
| **8** | 4.07 |
| **9** | 4.04 |
| **10** | 4.04 |

1. Consideration

The time difference from section 5.4.1 is observed only 660ms. The GFX Library usually take longer time for initialization before drawing the 1st frame so we are estimating that the GFX driver initialization start earlier than the time of “VM powered on”.







## Interrupt Time

### Delay time for interrupt

1. Description

Measure the interrupt performance on virtualized Linux using Cyclictest.

1. Precondition

* Measure on virtualized Linux (Type4) without other applications
* Use Cyclictest on terminal software.
* Verified 10 times and use the average as the result value.
* This environment is set kernel optimization (kernel\_opt).

1. How to measure
2. Login to Linux.

salvator-x login: root

1. Run the following command to change the directory.

root@salvator-x:~# cd tools

1. Run the following Cyclictest’s command.

root@salvator-x:~/tools# /cyclictest -n -p 70 -i 4 -l 100000 -q -a

After finishing a command, you will see the log like below.

Red square is results.

# /dev/cpu\_dma\_latency set to 0us

T: 0 ( 1733) P:70 I:4 C: 100000 Min: 3 Act: 4 Avg: 5 Max: 29

1. Run the step 2 process 10 seconds after the result is displayed.

Repeat this 9 times.

1. Result

Table 5‑24: Result

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Virtualized Linux (Type4)** | | |
|  | **Ave.(us)** | **Min.(us)** | **Max.(us)** |
|  | Ave 5 | Min 3 | Max 35 |
| **1** | 5.00 | 3.00 | 29.00 |
| **2** | 5.00 | 4.00 | 31.00 |
| **3** | 5.00 | 3.00 | 27.00 |
| **4** | 5.00 | 3.00 | 29.00 |
| **5** | 5.00 | 4.00 | 31.00 |
| **6** | 5.00 | 3.00 | 35.00 |
| **7** | 5.00 | 3.00 | 25.00 |
| **8** | 5.00 | 4.00 | 30.00 |
| **9** | 6.00 | 3.00 | 26.00 |
| **10** | 4.00 | 3.00 | 26.00 |

1. Consideration

This result is expected.

### Delay time variation

1. Description

Measure the maximum and minimum interrupt performance value on virtualized Linux using Cyclictest.

1. Precondition

* Extract maximum value and minimum value at 5.5.1 results.

1. How to measure
2. Type 4: Refer to 5.5.1.
3. Result

Table 5‑25: Result

|  |  |  |
| --- | --- | --- |
|  | **Min(us)** | **Max(us)** |
| **Virtualized Linux (Type4)** | 3 | 35 |

1. Consideration

This result is expected.

### Lock Synchronization latency

Out of Scope.



































































## RAM I/O Performance

### RAM I/O Performance

1. Description

Measure the RAM’s Read/Write performance when use virtualized Linux, native Linux, and native INTEGRITY.

Measurement tool is lmbench.

1. Precondition

* Measure on special native INTEGRITY for this measurement.
* Use porting lmbench for native INTEGRITY.
* Measure on virtualized Linux, native Linux, and native INTEGRITY (Type2, Type4 and Type3).
* Use lmbench for Linux.
* Verified 10 times and use the average as the result value.

1. How to measure

* Type2, Type4 using lmbench for Linux.

1. Login to Linux.

salvator-x login: root

1. Run the following command to change the directory.

root@salvator-x:~# cd tools/lmbench

1. Run the following command for measurement.

root@salvator-x:~/tools/lmbench# ./bw\_file\_rd 64M io\_only /home/root/test.txt

After finishing a command, you will see the log like below.

Red square is results.

root@salvator-x:~/tools/lmbench# ./bw\_file\_rd 64M io\_only /home/root/test.txt

64.00 1265.77

* Type3 using porting lmbench for native INTEGRITY

1. Lunch Type3 for this measure.
2. Select “Dynamic Download/INDRT Connection (rtserv2) for Device Tree” and press “Connect” button.
3. Select “Run mode target”
4. Right click on “create file”
5. Select [Target] - [Load Module] - [Load Module...] from Menu bar.
6. Load the " bw\_file\_rd.ael " file included in the deliverables.

You will see the log like below on cmd-tab.

Red square is results.

Target: Loader: Application started

I/O: 1073.7418 MB in 0.0004 secs, 3012743.6139 MB/sec

1. Result

Table 5‑21: Type3 Result (only posix kernel – T17.0)

|  |  |
| --- | --- |
|  | **Native INTEGRITY [MB/sec]** |
| **Ave.** | 3235054.0233 |
| **1** | 3172644.3553 |
| **2** | 3242780.4242 |
| **3** | 3243933.0030 |
| **4** | 3257246.7894 |
| **5** | 3231728.5899 |
| **6** | 3249455.1133 |
| **7** | 3210272.7766 |
| **8** | 3264820.4271 |
| **9** | 3225055.2251 |
| **10** | 3252603.5296 |











1. Consideration

The INTEGRITY’s result is unnaturally large. As this benchmark program (bw\_file\_rd in lmbench) just issue file read command and not actually use the file data, possibly the actual file read operation can be omitted.

The result of the native Linux is 1432.17MB/sec, and the virtualized Linux is 1270.79MB/sec. As the row memory read bandwidth that are measured in 5.11.1 are 2345.02MB/sec and 2338.82MB/sec, the results are reasonable. The performance of virtualized Linux is 11.27% slower than native one. As the filesystem operation usually require many synchronization in the OS domain, it is assumed that relatively larger performance overhead will appear.

## Memory Performance

### Sequential reading performance

1. Description

Measure the performance to read sequential blocks of memory on virtualized Linux and native Linux.

Measurement tool is lmbench.

1. Precondition

* Measure on virtualized Linux on virtualized Linux and native Linux (Type4 and Type2)

\*Both types stop the function of Linux App

* Use lmbench’s lat\_mem\_rd command on terminal software.
* Compare the performance between virtualized Linux and native Linux. The performance results should be near to native OS implementations results.
* Verified 10 times and use the average as the result value.

1. How to measure

* Memory read latency

1. Login to Linux.

salvator-x login: root

1. Run the following command to change the directory.

root@salvator-x:~# cd tools/lmbench

1. Run the following command for measurement.

root@salvator-x:~/tools/lmbench# ./lat\_mem\_rd 64m 64

After finishing a command, you will see the log like below.

Red square is results.

root@salvator-x:~/tools/lmbench# ./lat\_mem\_rd 64m 64

"stride=64

0.00049 2.671

0.00098 2.671

0.00195 2.671

0.00293 2.671

0.00391 2.671

0.00586 2.671

0.00781 2.671

0.01172 2.671

0.01562 5.235

0.02344 4.483

0.03125 5.310

0.04688 6.111

0.06250 5.699

0.09375 6.090

0.12500 6.074

0.18750 6.317

0.25000 6.314

0.37500 6.312

0.50000 6.311

0.75000 6.310

1.00000 6.255

1.50000 6.458

2.00000 35.175

3.00000 39.063

4.00000 31.825

6.00000 31.588

8.00000 31.409

12.00000 31.255

16.00000 31.164

24.00000 31.096

32.00000 31.037

48.00000 30.996

64.00000 31.199

1. Run the step 3 process 10 seconds after the result is displayed.

Repeat this 9 times.

* Measure Mbyte/s

1. Login to Linux.

salvator-x login: root

1. Run the following command to change the directory.

root@salvator-x:~# cd tools/lmbench

1. Run the following command for measurement.

root@salvator-x:~/tools/lmbench# ./bw\_mem 64m rd

After finishing a command, you will see the log like below.

Red square is results.

root@salvator-x:~/tools/lmbench# ./bw\_mem 64m rd

67.11 2346.46

1. Run the step 3 process 10 seconds after the result is displayed.

Repeat this 9 times.

1. Result

**Table 5‑45: Memory read latency Result (Type4)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Array size**  **(MB)** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **0.00049** | 2.67 | 2.67 | 2.67 | 2.68 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 |
| **0.00098** | 2.67 | 2.67 | 2.67 | 2.68 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 |
| **0.00195** | 2.67 | 2.67 | 2.67 | 2.68 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 |
| **0.00293** | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 |
| **0.00391** | 2.67 | 2.67 | 2.67 | 2.68 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 |
| **0.00586** | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 |
| **0.00781** | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 |
| **0.01172** | 2.67 | 2.67 | 2.67 | 2.68 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 |
| **0.01562** | 5.24 | 5.34 | 2.67 | 2.68 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 |
| **0.02344** | 4.48 | 2.67 | 4.42 | 4.38 | 2.67 | 4.52 | 4.50 | 2.67 | 2.67 | 2.67 |
| **0.03125** | 5.31 | 3.98 | 4.47 | 3.98 | 4.06 | 3.98 | 4.90 | 5.69 | 4.40 | 4.41 |
| **0.04688** | 6.11 | 5.80 | 5.02 | 5.01 | 5.22 | 5.84 | 5.80 | 5.26 | 5.80 | 4.43 |
| **0.06250** | 5.70 | 6.11 | 5.90 | 6.10 | 6.09 | 6.09 | 5.90 | 6.02 | 5.89 | 6.09 |
| **0.09375** | 6.09 | 6.07 | 6.09 | 6.07 | 6.07 | 6.07 | 5.50 | 6.07 | 6.07 | 6.08 |
| **0.12500** | 6.07 | 6.08 | 6.07 | 6.08 | 6.08 | 6.07 | 5.67 | 6.06 | 6.07 | 6.07 |
| **0.18750** | 6.32 | 6.33 | 6.32 | 6.33 | 6.32 | 6.32 | 6.32 | 6.31 | 6.33 | 6.32 |
| **0.25000** | 6.31 | 6.31 | 6.31 | 6.32 | 6.32 | 6.31 | 6.31 | 6.31 | 6.32 | 6.31 |
| **0.37500** | 6.31 | 6.31 | 6.31 | 6.31 | 6.31 | 6.31 | 6.31 | 6.31 | 6.31 | 6.31 |
| **0.50000** | 6.31 | 6.32 | 6.32 | 6.30 | 6.31 | 6.31 | 6.31 | 6.31 | 6.32 | 6.31 |
| **0.75000** | 6.31 | 6.31 | 6.31 | 6.11 | 6.31 | 6.31 | 6.31 | 6.31 | 6.31 | 6.31 |
| **1.00000** | 6.26 | 6.32 | 6.31 | 6.13 | 6.31 | 6.31 | 6.30 | 6.31 | 6.27 | 6.43 |
| **1.50000** | 6.46 | 12.61 | 11.93 | 7.09 | 8.19 | 13.29 | 9.64 | 10.34 | 9.71 | 8.16 |
| **2.00000** | 35.18 | 34.06 | 35.93 | 31.88 | 33.05 | 36.66 | 35.88 | 38.78 | 34.59 | 38.16 |
| **3.00000** | 39.06 | 37.51 | 42.61 | 37.27 | 43.93 | 43.61 | 38.87 | 40.44 | 39.81 | 41.78 |
| **4.00000** | 31.83 | 32.23 | 32.76 | 31.58 | 32.47 | 33.19 | 31.91 | 32.65 | 31.83 | 32.75 |
| **6.00000** | 31.59 | 31.98 | 32.18 | 31.36 | 32.03 | 32.45 | 31.65 | 32.12 | 31.73 | 32.14 |
| **8.00000** | 31.41 | 31.60 | 31.83 | 31.25 | 31.76 | 32.04 | 31.45 | 31.78 | 31.42 | 31.82 |
| **12.00000** | 31.26 | 31.49 | 31.54 | 31.26 | 31.52 | 31.66 | 31.27 | 31.49 | 31.26 | 31.50 |
| **16.00000** | 31.16 | 31.26 | 31.39 | 31.08 | 31.41 | 31.48 | 31.18 | 31.35 | 31.18 | 31.36 |
| **24.00000** | 31.10 | 31.15 | 31.21 | 31.03 | 31.28 | 31.30 | 31.10 | 31.20 | 31.08 | 31.21 |
| **32.00000** | 31.04 | 31.11 | 31.27 | 31.01 | 31.22 | 31.20 | 31.06 | 31.15 | 31.17 | 31.26 |
| **48.00000** | 31.00 | 31.03 | 31.09 | 30.98 | 31.16 | 31.11 | 31.01 | 31.07 | 31.00 | 31.07 |
| **64.00000** | 31.20 | 31.06 | 31.05 | 31.06 | 31.17 | 31.06 | 31.05 | 31.05 | 31.06 | 31.05 |

**Table 5‑46: Memory read latency Result (Type2)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Array size**  **(MB)** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **0.00049** | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.68 | 2.67 | 2.67 |
| **0.00098** | 2.67 | 2.67 | 2.68 | 2.67 | 2.67 | 2.67 | 2.67 | 2.68 | 2.67 | 2.67 |
| **0.00195** | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.68 | 2.68 | 2.67 | 2.67 |
| **0.00293** | 2.67 | 2.67 | 2.68 | 2.68 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 |
| **0.00391** | 2.67 | 2.68 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 |
| **0.00586** | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.68 | 2.67 | 2.67 |
| **0.00781** | 2.67 | 2.67 | 2.68 | 2.67 | 2.67 | 2.68 | 2.68 | 2.68 | 2.67 | 2.67 |
| **0.01172** | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 |
| **0.01562** | 2.67 | 2.67 | 2.68 | 2.67 | 2.67 | 5.37 | 5.51 | 5.37 | 2.67 | 2.67 |
| **0.02344** | 2.67 | 2.67 | 4.97 | 4.45 | 2.67 | 2.68 | 4.45 | 2.68 | 4.51 | 4.41 |
| **0.03125** | 4.10 | 3.98 | 4.00 | 5.24 | 2.67 | 4.45 | 4.46 | 4.48 | 2.67 | 3.98 |
| **0.04688** | 6.11 | 5.57 | 5.58 | 5.81 | 5.54 | 5.83 | 6.13 | 6.13 | 4.95 | 5.58 |
| **0.06250** | 5.25 | 6.09 | 6.09 | 5.70 | 5.70 | 5.29 | 5.70 | 5.27 | 5.25 | 5.86 |
| **0.09375** | 6.09 | 6.07 | 6.09 | 6.07 | 6.09 | 6.07 | 6.09 | 6.11 | 5.80 | 6.06 |
| **0.12500** | 6.07 | 6.07 | 6.07 | 6.08 | 6.07 | 6.10 | 6.07 | 6.10 | 5.88 | 6.07 |
| **0.18750** | 6.31 | 6.31 | 6.31 | 6.32 | 6.31 | 6.31 | 6.32 | 6.31 | 6.31 | 6.31 |
| **0.25000** | 6.31 | 6.31 | 6.34 | 6.31 | 6.31 | 6.31 | 6.34 | 6.34 | 6.31 | 6.31 |
| **0.37500** | 6.31 | 6.33 | 6.31 | 6.31 | 6.31 | 6.33 | 6.31 | 6.31 | 6.31 | 6.31 |
| **0.50000** | 6.30 | 6.31 | 6.33 | 6.33 | 6.31 | 6.30 | 6.33 | 6.33 | 6.31 | 6.30 |
| **0.75000** | 6.23 | 6.31 | 6.15 | 6.31 | 6.31 | 6.30 | 6.31 | 6.33 | 6.30 | 6.30 |
| **1.00000** | 6.18 | 6.30 | 6.14 | 6.51 | 6.31 | 6.30 | 6.32 | 6.33 | 6.30 | 6.30 |
| **1.50000** | 6.04 | 6.91 | 7.47 | 15.30 | 12.11 | 22.10 | 7.40 | 7.28 | 9.00 | 10.78 |
| **2.00000** | 27.18 | 32.63 | 27.14 | 35.95 | 30.11 | 31.23 | 33.97 | 33.97 | 39.46 | 34.08 |
| **3.00000** | 41.69 | 43.86 | 38.90 | 45.89 | 41.06 | 46.38 | 40.83 | 47.37 | 46.87 | 44.68 |
| **4.00000** | 31.56 | 33.43 | 31.49 | 32.83 | 32.41 | 32.44 | 32.43 | 33.53 | 33.22 | 33.18 |
| **6.00000** | 31.37 | 32.41 | 31.61 | 32.15 | 31.95 | 31.93 | 31.78 | 32.30 | 32.33 | 32.33 |
| **8.00000** | 31.21 | 32.00 | 31.25 | 31.86 | 31.59 | 31.95 | 31.74 | 32.23 | 31.90 | 31.89 |
| **12.00000** | 31.04 | 31.63 | 31.11 | 31.52 | 31.29 | 31.43 | 31.21 | 31.59 | 31.50 | 31.48 |
| **16.00000** | 30.97 | 31.30 | 31.06 | 31.58 | 31.16 | 31.30 | 31.42 | 31.30 | 31.28 | 31.31 |
| **24.00000** | 30.89 | 31.12 | 31.01 | 31.21 | 31.02 | 31.37 | 31.10 | 31.23 | 31.11 | 31.12 |
| **32.00000** | 30.88 | 31.02 | 31.17 | 31.31 | 30.95 | 31.10 | 31.00 | 31.15 | 31.01 | 31.03 |
| **48.00000** | 30.81 | 30.95 | 30.95 | 30.93 | 30.88 | 31.21 | 30.98 | 31.04 | 30.92 | 30.94 |
| **64.00000** | 30.88 | 31.22 | 31.21 | 31.21 | 30.87 | 31.00 | 31.22 | 30.99 | 30.87 | 31.00 |

**Table 5‑47: Memory read latency Result**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Virtualized Linux (Type4) [ns]** | | | **Native Linux(Type2) [ns]** | | |
| **Array size**  **(MB)** | **Ave.** | **Min.** | **Max.** | **Ave.** | **Min.** | **Max.** |
| **0.00049** | 2.67 | 2.67 | 2.68 | 2.67 | 2.67 | 2.68 |
| **0.00098** | 2.67 | 2.67 | 2.68 | 2.67 | 2.67 | 2.68 |
| **0.00195** | 2.67 | 2.67 | 2.68 | 2.67 | 2.67 | 2.68 |
| **0.00293** | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.68 |
| **0.00391** | 2.67 | 2.67 | 2.68 | 2.67 | 2.67 | 2.68 |
| **0.00586** | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.68 |
| **0.00781** | 2.67 | 2.67 | 2.67 | 2.67 | 2.67 | 2.68 |
| **0.01172** | 2.67 | 2.67 | 2.68 | 2.67 | 2.67 | 2.67 |
| **0.01562** | 3.20 | 2.67 | 5.34 | 3.49 | 2.67 | 5.51 |
| **0.02344** | 3.57 | 2.67 | 4.52 | 3.62 | 2.67 | 4.97 |
| **0.03125** | 4.52 | 3.98 | 5.69 | 4.00 | 2.67 | 5.24 |
| **0.04688** | 5.43 | 4.43 | 6.11 | 5.72 | 4.95 | 6.13 |
| **0.06250** | 5.99 | 5.70 | 6.11 | 5.62 | 5.25 | 6.09 |
| **0.09375** | 6.02 | 5.50 | 6.09 | 6.05 | 5.80 | 6.11 |
| **0.12500** | 6.03 | 5.67 | 6.08 | 6.06 | 5.88 | 6.10 |
| **0.18750** | 6.32 | 6.31 | 6.33 | 6.31 | 6.31 | 6.32 |
| **0.25000** | 6.31 | 6.31 | 6.32 | 6.32 | 6.31 | 6.34 |
| **0.37500** | 6.31 | 6.31 | 6.31 | 6.31 | 6.31 | 6.33 |
| **0.50000** | 6.31 | 6.30 | 6.32 | 6.32 | 6.30 | 6.33 |
| **0.75000** | 6.29 | 6.11 | 6.31 | 6.28 | 6.15 | 6.33 |
| **1.00000** | 6.29 | 6.13 | 6.43 | 6.30 | 6.14 | 6.51 |
| **1.50000** | 9.74 | 6.46 | 13.29 | 10.44 | 6.04 | 22.10 |
| **2.00000** | 35.42 | 31.88 | 38.78 | 32.57 | 27.14 | 39.46 |
| **3.00000** | 40.49 | 37.27 | 43.93 | 43.75 | 38.90 | 47.37 |
| **4.00000** | 32.32 | 31.58 | 33.19 | 32.65 | 31.49 | 33.53 |
| **6.00000** | 31.92 | 31.36 | 32.45 | 32.02 | 31.37 | 32.41 |
| **8.00000** | 31.64 | 31.25 | 32.04 | 31.76 | 31.21 | 32.23 |
| **12.00000** | 31.42 | 31.26 | 31.66 | 31.38 | 31.04 | 31.63 |
| **16.00000** | 31.29 | 31.08 | 31.48 | 31.27 | 30.97 | 31.58 |
| **24.00000** | 31.17 | 31.03 | 31.30 | 31.12 | 30.89 | 31.37 |
| **32.00000** | 31.15 | 31.01 | 31.27 | 31.06 | 30.88 | 31.31 |
| **48.00000** | 31.05 | 30.98 | 31.16 | 30.96 | 30.81 | 31.21 |
| **64.00000** | 31.08 | 31.05 | 31.20 | 31.05 | 30.87 | 31.22 |

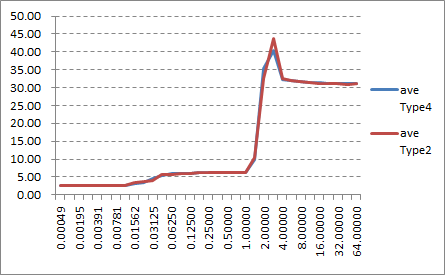
****

Figure 5‑13: Average Result Linux and Native Linux (Type4 and Type2)

**Table 5‑48: Data Transmission rate (Mbyte/s)**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test environment** | Ave. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **Virtualized Linux (Type4)** | 2338.82 | 2346.46 | 2338.45 | 2333.09 | 2334.79 | 2346.46 | 2336.09 | 2339.27 | 2338.21 | 2345.56 | 2329.85 |
| **Native Linux (Type2)** | 2345.02 | 2332.60 | 2347.61 | 2357.18 | 2332.68 | 2356.93 | 2332.44 | 2348.19 | 2348.35 | 2347.53 | 2346.71 |

1. Consideration

The result of "lat\_mem\_rd" reflects well about the memory system hierarchy of the target hardware.

Both Native Linux and Virtualized Linux has the similar form which has the 3 specific score groups.

The first one is around 2.7ns and mainly observed under 32kB test size.

In these cases, the read access of this test is cached in the main CPU's L1 data cache (ARM Cortex-A57: 32kB L1 data cache per core), so this reflects the L1 cache read performance.

The next one is about 6.0ns to 6.3ns and mainly observed between 32kB to 1.5MB.

In these cases, the read access size exceeds the L1 data cache size and causes L1 data cache miss, but can be cached in the 2MB L2 unified cache, so this reflects the L2 cache read performance.

And the last one is around 31ns and mainly observed in 2MB or more.

In this size, the read access of this test exceeds the L2 cache size and causes L2 cache miss every time once the L2 cache is fully filled by the test read.

As L2 cache miss takes cache line fetch from the main memory (LPDDR4 memory on R-Car H3 SiP), this reflects the LPDDR4 read performance.

It is observed that the latency time of virtualized Linux increases a little earlier than native Linux when the test size increases. For this behavior, we are estimating that this is effected by the program working set size difference of the operating system portion. The virtualized Linux requires OS kernel feature not only for Linux but also for INTEGRITY kernel, and consumes more cache lines for operation. When the test size is nearly equal to either L1 or L2 cachesize, it increases the virtualized Linux’s latency time.

### Sequential writing performance

1. Description

Measure the performance to write sequential blocks of memory on virtualized Linux and native Linux.

Measurement tool is lmbench.

1. Precondition

* Measure on virtualized Linux on virtualized Linux and native Linux (Type4 and Type2)

\*Both types stop the function of Linux App

* Use lmbench’s lat\_mem\_wr command on terminal software.
* Compare the performance between virtualized Linux and native Linux. The performance results should be near to native OS implementations results.
* Verified 10 times and use the average as the result value.

1. How to measure
2. Login to Linux.

salvator-x login: root

1. Run the following command to change the directory.

root@salvator-x:~# cd tools/lmbench

1. Run the following command for measurement.

root@salvator-x:~# cd tools/lmbench

root@salvator-x:~/tools/lmbench# ./bw\_mem 1m wr

root@salvator-x:~/tools/lmbench# ./bw\_mem 2m wr

root@salvator-x:~/tools/lmbench# ./bw\_mem 4m wr

root@salvator-x:~/tools/lmbench# ./bw\_mem 8m wr

root@salvator-x:~/tools/lmbench# ./bw\_mem 16m wr

root@salvator-x:~/tools/lmbench# ./bw\_mem 32m wr

root@salvator-x:~/tools/lmbench# ./bw\_mem 64m wr

After finishing a command, you will see the log like below.

Red square is results.

root@salvator-x:~/tools/lmbench# ./bw\_mem 1m wr

1.05 13676.19

1. Run the step 3 process 10 seconds after the result is displayed.

Repeat this 9 times.

1. Result

**Table 5‑49: Result (Type4)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test size | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **1M** | 13609.71 | 13669.76 | 13666.27 | 13660.88 | 13666.27 | 13672.12 | 13570.52 | 13627.70 | 13664.70 | 13664.70 |
| **2M** | 3913.92 | 4018.93 | 4286.26 | 4491.49 | 4316.61 | 4146.06 | 4414.28 | 4215.77 | 4917.56 | 3799.19 |
| **4M** | 2211.02 | 2202.89 | 2221.95 | 2209.47 | 2209.08 | 2212.19 | 2215.30 | 2216.47 | 2195.59 | 2203.66 |
| **8M** | 1952.20 | 1959.50 | 1962.25 | 1952.88 | 1952.20 | 1950.16 | 1958.58 | 1958.35 | 1950.16 | 1953.79 |
| **16M** | 1938.89 | 1943.61 | 1946.54 | 1939.56 | 1942.26 | 1942.93 | 1944.73 | 1937.55 | 1940.01 | 1938.44 |
| **32M** | 1940.80 | 1928.75 | 1940.68 | 1939.45 | 1947.44 | 1945.64 | 1946.76 | 1940.35 | 1948.91 | 1947.10 |
| **64M** | 1944.56 | 1940.80 | 1948.06 | 1946.93 | 1947.10 | 1941.70 | 1938.67 | 1940.80 | 1940.96 | 1940.85 |

**Table 5‑50: Result (Type2)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test size | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **1M** | 13676.19 | 13633.49 | 13676.19 | 13631.28 | 13680.58 | 13674.38 | 13671.91 | 13670.09 | 13609.71 | 13699.20 |
| **2M** | 4759.94 | 5047.76 | 5493.01 | 5387.18 | 4085.12 | 5110.21 | 4948.81 | 4822.74 | 4979.54 | 4768.96 |
| **4M** | 2237.36 | 2236.56 | 2221.17 | 2248.55 | 2220.38 | 2235.37 | 2218.03 | 2234.98 | 2231.01 | 2231.80 |
| **8M** | 1971.01 | 1966.39 | 1951.97 | 1967.77 | 1969.85 | 1964.09 | 1975.42 | 1972.40 | 1947.44 | 1968.46 |
| **16M** | 1949.71 | 1952.66 | 1952.20 | 1951.75 | 1953.34 | 1952.43 | 1957.44 | 1952.20 | 1951.97 | 1953.11 |
| **32M** | 1955.04 | 1935.65 | 1950.05 | 1955.27 | 1951.52 | 1954.13 | 1953.00 | 1954.82 | 1935.87 | 1936.09 |
| **64M** | 1937.66 | 1951.92 | 1950.67 | 1952.31 | 1952.37 | 1952.03 | 1952.20 | 1955.04 | 1936.20 | 1953.96 |

Table 5‑51: Result

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test size | **Virtualized Linux (Type4) [ns]** | | | **Native Linux(Type2) [ns]** | | |
|  | **Ave.** | **Min.** | **Max.** | **Ave.** | **Min.** | **Max.** |
| **1M** | 13647.26 | 13570.52 | 13672.12 | 13662.30 | 13609.71 | 13699.20 |
| **2M** | 4252.01 | 3799.19 | 4917.56 | 4940.33 | 4085.12 | 5493.01 |
| **4M** | 2209.76 | 2195.59 | 2221.95 | 2231.52 | 2218.03 | 2248.55 |
| **8M** | 1955.01 | 1950.16 | 1962.25 | 1965.48 | 1947.44 | 1975.42 |
| **16M** | 1941.45 | 1937.55 | 1946.54 | 1952.68 | 1949.71 | 1957.44 |
| **32M** | 1942.59 | 1928.75 | 1948.91 | 1948.14 | 1935.65 | 1955.27 |
| **64M** | 1943.04 | 1938.67 | 1948.06 | 1949.44 | 1936.20 | 1955.04 |

1. Consideration

As lmbench doesn’t has the benchmark for the memory write latency, this time we used memory bandwidth benchmark instead. As the cache line length is 64 bytes, the bandwidth of 1938.67MB/sec (64MB on Type4) and 1936.20MB/sec (64MB on Type2) are corresponding to 33.012ns and 33.054ns of the latency time. They are a little larger than read latency, and this is reasonable because the write cache miss initially issues memory read to fill the cache line, and replace the specific position of the cache line by the store data.

### Random reading performance

1. Description

Measure the performance to read random blocks of memory on virtualized Linux and native Linux.

Measurement tool is lmbench.

1. Precondition

* Measure on virtualized Linux on virtualized Linux and native Linux (Type4 and Type2)

\*Both types stop the function of Linux App

* Use lmbench’s lat\_mem\_rd command on terminal software.
* Compare the performance between virtualized Linux and native Linux. The performance results should be near to native OS implementations results.
* Verified 10 times and use the average as the result value.

1. How to measure
2. Login to Linux.

salvator-x login: root

1. Run the following command to change the directory.

root@salvator-x:~# cd tools/lmbench

1. Run the following command for measurement.

root@salvator-x:~/tools/lmbench# ./lat\_mem\_rd 64m 131072

After finishing a command, you will see the log like below.

Red square is results.

root@salvator-x:~/tools/lmbench# ./lat\_mem\_rd 64m 131072

"stride=131072

0.12500 2.669

0.18750 2.669

0.25000 2.669

0.37500 2.669

0.50000 7.675

0.75000 11.827

1.00000 13.012

1.50000 14.905

2.00000 15.180

3.00000 85.090

4.00000 124.127

6.00000 150.806

8.00000 161.898

12.00000 174.101

16.00000 183.324

24.00000 203.268

32.00000 219.112

48.00000 316.886

64.00000 324.883

4. Run the step 3 process 10 seconds after the result is displayed.

Repeat this 9 times.

1. Result

**Table 5‑52: Result (Type4)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Array size**  **(MB)** | **1** | **2** | **3** | **4** | **5.** | **6** | **7** | **8** | **9** | **10** |
| **0.1250** | 2.678 | 2.677 | 2.678 | 2.671 | 2.678 | 2.678 | 2.678 | 2.674 | 2.674 | 2.677 |
| **0.1875** | 2.677 | 2.675 | 2.671 | 2.671 | 2.674 | 2.675 | 2.675 | 2.675 | 2.675 | 2.674 |
| **0.2500** | 2.675 | 2.671 | 2.674 | 2.671 | 2.671 | 2.671 | 2.671 | 2.671 | 2.675 | 2.674 |
| **0.3750** | 2.671 | 7.589 | 2.671 | 2.671 | 2.675 | 2.675 | 2.675 | 2.671 | 2.671 | 2.675 |
| **0.5000** | 2.675 | 7.681 | 2.675 | 2.67 | 14.126 | 2.675 | 2.671 | 2.671 | 2.674 | 2.671 |
| **0.7500** | 2.671 | 7.635 | 2.675 | 2.671 | 14.154 | 2.671 | 2.675 | 2.674 | 2.671 | 14.045 |
| **1.0000** | 2.675 | 7.261 | 9.692 | 7.154 | 13.04 | 6.062 | 9.925 | 7.154 | 10.872 | 9.148 |
| **1.5000** | 11.053 | 10.047 | 10.351 | 14.08 | 13.86 | 12.149 | 11.982 | 12.132 | 11.033 | 12.133 |
| **2.0000** | 13.015 | 12.4 | 11.603 | 13.443 | 15.215 | 13.109 | 12.065 | 12.746 | 13.71 | 13.249 |
| **3.0000** | 14.612 | 13.876 | 25.972 | 22.967 | 100.178 | 13.767 | 51.394 | 13.765 | 14.156 | 53.043 |
| **4.0000** | 14.942 | 33.825 | 87.631 | 86.079 | 126.024 | 50.151 | 103.167 | 42.234 | 17.947 | 103.829 |
| **6.0000** | 95.295 | 104.702 | 127.579 | 129.772 | 150.442 | 103.938 | 134.868 | 103.974 | 95.932 | 138.147 |
| **8.0000** | 124.612 | 127.143 | 143.325 | 144.049 | 159.85 | 130.508 | 148.273 | 129.189 | 122.214 | 151.129 |
| **12.0000** | 145.637 | 145.644 | 157.804 | 157.374 | 169.459 | 147.583 | 163.278 | 149.143 | 145.502 | 165.212 |
| **16.0000** | 158.973 | 157.879 | 167.848 | 167.535 | 176.399 | 160.876 | 171.76 | 160.594 | 156.41 | 172.546 |
| **24.0000** | 174.125 | 173.915 | 180.59 | 179.913 | 186.548 | 175.466 | 181.717 | 174.025 | 173.975 | 183.141 |
| **32.0000** | 189.616 | 190.056 | 193.498 | 194.073 | 199.172 | 191.493 | 195.995 | 190.194 | 189.508 | 196.876 |
| **48.0000** | 201.914 | 203.956 | 211.815 | 210.206 | 218.474 | 204.41 | 212.537 | 203.416 | 202.865 | 213.208 |
| **64.0000** | 259.192 | 260.523 | 260.651 | 258.855 | 260.39 | 260.431 | 258.424 | 260.634 | 256.988 | 259.821 |

**Table 5‑53: Result (Type2)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Array size**  **(MB)** | **1** | **2** | **3** | **4** | **5.** | **6** | **7** | **8** | **9** | **10** |
| **0.1250** | 2.685 | 2.669 | 2.685 | 2.669 | 2.669 | 2.685 | 2.685 | 2.669 | 2.685 | 2.684 |
| **0.1875** | 2.685 | 2.685 | 2.685 | 2.669 | 2.669 | 2.685 | 2.669 | 2.669 | 2.669 | 2.669 |
| **0.2500** | 2.685 | 2.669 | 2.669 | 2.669 | 2.669 | 2.685 | 2.685 | 2.669 | 2.685 | 2.685 |
| **0.3750** | 2.685 | 2.685 | 2.685 | 2.669 | 2.669 | 7.717 | 2.669 | 2.669 | 2.685 | 2.685 |
| **0.5000** | 2.685 | 7.673 | 2.669 | 2.669 | 2.669 | 14.89 | 2.685 | 2.669 | 7.673 | 2.685 |
| **0.7500** | 7.618 | 14.457 | 10.907 | 2.669 | 7.618 | 16.108 | 2.685 | 7.618 | 12.976 | 5.07 |
| **1.0000** | 6.084 | 14.934 | 9.925 | 11.321 | 10.426 | 16.108 | 10.571 | 14.513 | 12.836 | 2.685 |
| **1.5000** | 12.585 | 15.513 | 13.424 | 11.566 | 11.176 | 15.771 | 11.232 | 11.232 | 13.759 | 9.731 |
| **2.0000** | 13.047 | 15.395 | 12.124 | 14.013 | 12.972 | 15.86 | 13.089 | 12.805 | 14.346 | 12.459 |
| **3.0000** | 14.846 | 94.259 | 40.594 | 14.599 | 14.596 | 112.466 | 14.934 | 14.68 | 73.577 | 14.431 |
| **4.0000** | 50.366 | 128.609 | 95.775 | 24.694 | 24.349 | 135.323 | 50.15 | 49.882 | 116.946 | 72.706 |
| **6.0000** | 112.184 | 149.779 | 132.711 | 98.118 | 98.052 | 153.704 | 110.25 | 102.457 | 142.839 | 117.665 |
| **8.0000** | 133.017 | 156.419 | 144.157 | 123.454 | 121.296 | 159.388 | 131.707 | 127.372 | 151.527 | 134.726 |
| **12.0000** | 146.128 | 162.947 | 154.907 | 139.549 | 139.507 | 166.064 | 145.149 | 143.174 | 160.646 | 148.956 |
| **16.0000** | 154.18 | 167.999 | 161.478 | 150.347 | 149.064 | 169.785 | 154.012 | 154.19 | 165.46 | 156.009 |
| **24.0000** | 163.744 | 172.576 | 169.516 | 160.806 | 160.214 | 173.623 | 163.827 | 163.545 | 171.947 | 166.264 |
| **32.0000** | 171.928 | 179.99 | 175.803 | 169.935 | 171.108 | 180.781 | 171.61 | 170.506 | 179.332 | 174.417 |
| **48.0000** | 182.319 | 187.053 | 185.623 | 180.483 | 181.039 | 188.694 | 182.848 | 181.476 | 187.615 | 184.219 |
| **64.0000** | 219.294 | 221.76 | 219.582 | 221.968 | 219.059 | 227.985 | 219.693 | 220.078 | 221.206 | 221.885 |

Table 5‑54: Result

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Virtualized Linux (Type4) [ns]** | | | **Native Linux(Type2) [ns]** | | |
| **Array size**  **(MB)** | **Ave.** | **Min.** | **Max.** | **Ave.** | **Min.** | **Max.** |
| **0.1250** | 2.68 | 2.67 | 2.68 | 2.68 | 2.67 | 2.69 |
| **0.1875** | 2.67 | 2.67 | 2.68 | 2.68 | 2.67 | 2.69 |
| **0.2500** | 2.67 | 2.67 | 2.68 | 2.68 | 2.67 | 2.69 |
| **0.3750** | 3.16 | 2.67 | 7.59 | 3.18 | 2.67 | 7.72 |
| **0.5000** | 4.32 | 2.67 | 14.13 | 4.90 | 2.67 | 14.89 |
| **0.7500** | 5.45 | 2.67 | 14.15 | 8.77 | 2.67 | 16.11 |
| **1.0000** | 8.30 | 2.68 | 13.04 | 10.94 | 2.69 | 16.11 |
| **1.5000** | 11.88 | 10.05 | 14.08 | 12.60 | 9.73 | 15.77 |
| **2.0000** | 13.06 | 11.60 | 15.22 | 13.61 | 12.12 | 15.86 |
| **3.0000** | 32.37 | 13.77 | 100.18 | 40.90 | 14.43 | 112.47 |
| **4.0000** | 66.58 | 14.94 | 126.02 | 74.88 | 24.35 | 135.32 |
| **6.0000** | 118.46 | 95.30 | 150.44 | 121.78 | 98.05 | 153.70 |
| **8.0000** | 138.03 | 122.21 | 159.85 | 138.31 | 121.30 | 159.39 |
| **12.0000** | 154.66 | 145.50 | 169.46 | 150.70 | 139.51 | 166.06 |
| **16.0000** | 165.08 | 156.41 | 176.40 | 158.25 | 149.06 | 169.79 |
| **24.0000** | 178.34 | 173.92 | 186.55 | 166.61 | 160.21 | 173.62 |
| **32.0000** | 193.05 | 189.51 | 199.17 | 174.54 | 169.94 | 180.78 |
| **48.0000** | 208.28 | 201.91 | 218.47 | 184.14 | 180.48 | 188.69 |
| **64.0000** | 259.59 | 256.99 | 260.65 | 221.25 | 219.06 | 227.99 |

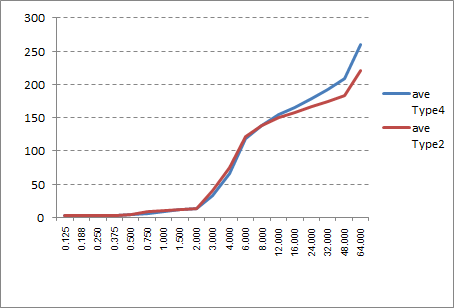


Figure 5‑14: Average Result Linux and Native Linux (Type4 and Type2)

1. Consideration

As it is not easy to perform 'random' access performance to the memory, this test simulates the random access condition by mostly-cache-miss address condition. In R-Car H3, the main CPU (ARM Cortex-A57) has 2MB unified L2C with 16-way and 64-byte line length. This means when the cacheable memory access is issued in every top boundary of 128kB, the top line of all 16 ways of cache arrays are consumed after 16th memory access, and all of the following memory access causes cache miss.

It is observed that when the test size is smaller than 8MB, the result is similar, or virtualized Linux is faster than native Linux. The faster virtualized Linux is currently unexpected and require further investigation.

In contrast, when the test size is larger than 8MB, the virtualized Linux is meaningfully slower than native Linux. As the TLB miss (both L1 data TLB and L2 unified TLB miss) occurs when the test size is larger than 4MB, the performance difference is expected to be produced by the difference of TLB miss processing. When in the native Linux environment, TLB miss causes the translation table walk once, but in virtualized Linux it causes twice due to the 2-stage address translation. See 5.11.7 and 5.11.8 for detail.

### Random writing performance

1. Description

Measure the performance to write random blocks of memory on virtualized Linux and native Linux.

Measurement tool is Renesas original test program.

1. Precondition

* Measure on virtualized Linux on virtualized Linux and native Linux (Type4 and Type2)

\*Both types stop the function of Linux App

* Use Renesas original test program. (random\_write)
* Compare the performance between virtualized Linux and native Linux. The performance results should be near to native OS implementations results.
* Verified 10 times and use the average as the result value.

1. How to measure
2. Login to Linux.

salvator-x login: root

1. Run the following command to change the directory.

root@salvator-x:~# cd tools

1. Run the following command to measure the Random writing performance 10 times.

root@salvator-x:~/tools# ./random\_write

After finishing a command, you will see the log like below.

Red square is results.

root@salvator-x:~/tools# ./random\_write

alloc\_size[byte], count, count\_max, start[sec], start[nsec], end[sec], end[nsec]

2097152, 0, 10, 184, 162289764, 184, 162338604, 48840

2097152, 1, 10, 184, 162434604, 184, 162472524, 37920

2097152, 2, 10, 184, 162481884, 184, 162482364, 480

2097152, 3, 10, 184, 162488124, 184, 162488604, 480

2097152, 4, 10, 184, 162494124, 184, 162494484, 360

2097152, 5, 10, 184, 162499884, 184, 162500364, 480

2097152, 6, 10, 184, 162505884, 184, 162506244, 360

2097152, 7, 10, 184, 162511524, 184, 162511884, 360

2097152, 8, 10, 184, 162517164, 184, 162517644, 480

2097152, 9, 10, 184, 162522804, 184, 162523164, 360

4194304, 0, 10, 184, 162541884, 184, 163453644, 911760

4194304, 1, 10, 184, 163559004, 184, 163601364, 42360

4194304, 2, 10, 184, 163608804, 184, 163612524, 3720

4194304, 3, 10, 184, 163618404, 184, 163621884, 3480

4194304, 4, 10, 184, 163627284, 184, 163628364, 1080

4194304, 5, 10, 184, 163633524, 184, 163634364, 840

4194304, 6, 10, 184, 163639644, 184, 163640364, 720

4194304, 7, 10, 184, 163645764, 184, 163646484, 720

4194304, 8, 10, 184, 163651644, 184, 163652364, 720

4194304, 9, 10, 184, 163657644, 184, 163658364, 720

8388608, 0, 10, 184, 163677084, 184, 165532045, 1854961

8388608, 1, 10, 184, 165683485, 184, 166216525, 533040

8388608, 2, 10, 184, 166231525, 184, 166237165, 5640

8388608, 3, 10, 184, 166243045, 184, 166246165, 3120

8388608, 4, 10, 184, 166251685, 184, 166254325, 2640

8388608, 5, 10, 184, 166259605, 184, 166262245, 2640

8388608, 6, 10, 184, 166267765, 184, 166270405, 2640

8388608, 7, 10, 184, 166275805, 184, 166278325, 2520

8388608, 8, 10, 184, 166283965, 184, 166286245, 2280

8388608, 9, 10, 184, 166292125, 184, 166294765, 2640

16777216, 0, 10, 184, 166320925, 184, 172131925, 5811000

16777216, 1, 10, 184, 172387765, 184, 174471565, 2083800

16777216, 2, 10, 184, 174489325, 184, 174509965, 20640

16777216, 3, 10, 184, 174516085, 184, 174525325, 9240

16777216, 4, 10, 184, 174531085, 184, 174539605, 8520

16777216, 5, 10, 184, 174545245, 184, 174553525, 8280

16777216, 6, 10, 184, 174559165, 184, 174566965, 7800

16777216, 7, 10, 184, 174572605, 184, 174581485, 8880

16777216, 8, 10, 184, 174587005, 184, 174595645, 8640

16777216, 9, 10, 184, 174601045, 184, 174608845, 7800

33554432, 0, 10, 184, 174636325, 184, 185291005, 10654680

33554432, 1, 10, 184, 185696725, 184, 197594606, 11897881

33554432, 2, 10, 184, 198046166, 184, 207793166, 9747000

33554432, 3, 10, 184, 208227446, 184, 221096247, 12868801

33554432, 4, 10, 184, 221500527, 184, 233233767, 11733240

33554432, 5, 10, 184, 233638287, 184, 244573288, 10935001

33554432, 6, 10, 184, 245018488, 184, 256515928, 11497440

33554432, 7, 10, 184, 256938688, 184, 268807409, 11868721

33554432, 8, 10, 184, 269193449, 184, 279993569, 10800120

33554432, 9, 10, 184, 280422569, 184, 290886090, 10463521

50331648, 0, 10, 184, 291260970, 184, 308172330, 16911360

50331648, 1, 10, 184, 308732370, 184, 325567291, 16834921

50331648, 2, 10, 184, 326156611, 184, 342395252, 16238641

50331648, 3, 10, 184, 343343972, 184, 359209892, 15865920

50331648, 4, 10, 184, 360752372, 184, 378412053, 17659681

50331648, 5, 10, 184, 378997413, 184, 395120494, 16123081

50331648, 6, 10, 184, 395674654, 184, 411530734, 15856080

50331648, 7, 10, 184, 412315774, 184, 429724175, 17408401

50331648, 8, 10, 184, 430367855, 184, 448244856, 17877001

50331648, 9, 10, 184, 448877616, 184, 465900817, 17023201

67108864, 0, 10, 184, 466462537, 184, 489351938, 22889401

67108864, 1, 10, 184, 491128778, 184, 514392819, 23264041

67108864, 2, 10, 184, 515150979, 184, 538065940, 22914961

67108864, 3, 10, 184, 538811260, 184, 561737380, 22926120

67108864, 4, 10, 184, 562498421, 184, 584818901, 22320480

67108864, 5, 10, 184, 585567341, 184, 607276542, 21709201

67108864, 6, 10, 184, 607982022, 184, 632766823, 24784801

67108864, 7, 10, 184, 633606943, 184, 657430784, 23823841

67108864, 8, 10, 184, 658176824, 184, 682881945, 24705121

67108864, 9, 10, 184, 683602905, 184, 706038106, 22435201

-- finished.

1. Result

**Table 5‑55: Result (Type4)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Array size**  **(MB)** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **2M** | 0.04236 | 0.02928 | 0.00072 | 0.00036 | 0.00036 | 0.00036 | 0.00036 | 0.00036 | 0.00036 | 0.00048 |
| **4M** | 0.65052 | 0.03360 | 0.00276 | 0.00120 | 0.00072 | 0.00072 | 0.00072 | 0.00072 | 0.00072 | 0.00072 |
| **8M** | 1.92516 | 0.60564 | 0.00420 | 0.00396 | 0.00216 | 0.00264 | 0.00204 | 0.00216 | 0.00228 | 0.00204 |
| **16M** | 4.38828 | 1.90884 | 0.01512 | 0.00708 | 0.00660 | 0.00756 | 0.00804 | 0.00780 | 0.00684 | 0.00684 |
| **32M** | 9.27384 | 8.73480 | 8.98452 | 9.02604 | 9.01296 | 9.14568 | 8.94012 | 9.05112 | 8.95584 | 8.90136 |
| **48M** | 13.67736 | 13.82160 | 13.53372 | 13.63812 | 13.56264 | 13.97652 | 14.03928 | 13.80252 | 13.91436 | 13.74948 |
| **64M** | 18.93768 | 17.70492 | 18.25560 | 17.74824 | 18.28896 | 17.91444 | 18.00984 | 17.75580 | 18.18672 | 18.36936 |

**Table 5‑56: Result (Type2)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Array size**  **(MB)** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **2M** | 0.03756 | 0.00192 | 0.00060 | 0.00072 | 0.00036 | 0.00048 | 0.00036 | 0.00072 | 0.00060 | 0.00036 |
| **4M** | 0.65796 | 0.00444 | 0.00228 | 0.00228 | 0.00216 | 0.00228 | 0.00180 | 0.00204 | 0.00216 | 0.00228 |
| **8M** | 1.86924 | 0.47436 | 0.00768 | 0.00504 | 0.00528 | 0.00576 | 0.00480 | 0.00504 | 0.00552 | 0.00540 |
| **16M** | 4.37940 | 1.61028 | 0.02004 | 0.01044 | 0.01044 | 0.01044 | 0.01044 | 0.01044 | 0.01032 | 0.01044 |
| **32M** | 9.33072 | 8.56212 | 8.75352 | 8.93064 | 9.02640 | 8.97372 | 9.09624 | 9.02280 | 9.07680 | 8.67060 |
| **48M** | 13.89948 | 12.71736 | 13.64856 | 13.78860 | 13.88268 | 13.87536 | 13.67964 | 13.75200 | 13.82088 | 14.01768 |
| **64M** | 17.94108 | 17.47488 | 17.56656 | 18.25212 | 18.69636 | 18.13944 | 18.29844 | 18.31908 | 17.66844 | 18.49116 |

Table 5‑57: Result

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Array size**  **(MB)** | **Virtualized Linux (Type4) [ns]** | | | **Native Linux (Type2) [ns]** | | |
| **Ave.** | **Min.** | **Max.** | **Ave.** | **Min.** | **Max.** |
| **2M** | 0.04236 | 0.00036 | 0.00750 | 0.03756 | 0.00036 | 0.004368 |
| **4M** | 0.65052 | 0.00072 | 0.06924 | 0.65796 | 0.0018 | 0.067968 |
| **8M** | 1.92516 | 0.00204 | 0.25523 | 1.86924 | 0.0048 | 0.238812 |
| **16M** | 4.38828 | 0.00660 | 0.63630 | 4.3794 | 0.01032 | 0.608268 |
| **32M** | 9.27384 | 8.73480 | 9.00263 | 9.33072 | 8.562121 | 8.9443564 |
| **48M** | 14.03928 | 13.53372 | 13.77156 | 14.01768 | 12.717361 | 13.7082245 |
| **64M** | 18.93768 | 17.70492 | 18.11716 | 18.696361 | 17.474881 | 18.0847568 |

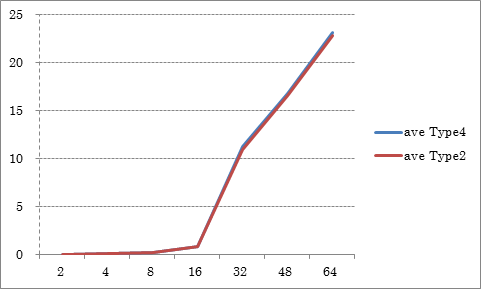


Figure 5‑15: Average Result Virtualized Linux and Native Linux (Type4 and Type2)

1. Consideration

As same as 5.11.3, this test also simulates the random access by using specific address condition to force the cache line conflict.

Generally, the test result (ms) is in proportional when the test size is larger than 8MB. The result for less or equal 8MB are vary. We are estimating that if the test size is smaller or equal to 8MB (= only 64 addresses for test), the store request will possibly merged on the store buffer and reduces the write requests for the LPDDR4 main memory. Unlike the random read case, the performance of the native Linux and virtualized Linux are observed almost same in the all test size. The possible reason is estimated that the store request can be buffered, and the TLB miss latency can be hidden in the preceding cache miss processing time.

### Memory Allocate/Deallocate performance

1. Description

Measure the performance to allocate blocks of memory and deallocate the same blocks on virtualized Linux and native Linux.

Measurement tool is Renesas original test program.

1. Precondition

* Measure on virtualized Linux on virtualized Linux and native Linux (Type4 and Type2)

\*Both types stop the function of Linux App

* Use Renesas original test program. (alloc\_dealloc)
* Compare the performance between virtualized Linux and native Linux. The performance results should be near to native OS implementations results.
* Verified 10 times and use the average as the result value.

1. How to measure
2. Login to Linux.

salvator-x login: root

1. Run the following command to change the directory.

root@salvator-x:~# cd tools

1. Run the following command to measure the Memory Allocate/Deallocate performance 10 times.

root@salvator-x:~/tools# ./alloc\_dealloc

After finishing a command, you will see the log like below.

Red square is results.

root@salvator-x:~/tools# ./alloc\_dealloc

alloc\_size[byte], count, count\_max, start[sec], start[nsec], end[sec], end[nsec], diff[nsec]

2097152, 0, 10, 119, 675161486, 119, 676956686, 1795200

2097152, 1, 10, 119, 676999046, 119, 678334646, 1335600

2097152, 2, 10, 119, 678348446, 119, 678367286, 18840

2097152, 3, 10, 119, 678373286, 119, 678384326, 11040

2097152, 4, 10, 119, 678389606, 119, 678399326, 9720

2097152, 5, 10, 119, 678404606, 119, 678414686, 10080

2097152, 6, 10, 119, 678420686, 119, 678430526, 9840

2097152, 7, 10, 119, 678435806, 119, 678445166, 9360

2097152, 8, 10, 119, 678450446, 119, 678459806, 9360

2097152, 9, 10, 119, 678465206, 119, 678474806, 9600

4194304, 0, 10, 119, 678480566, 119, 680846246, 2365680

4194304, 1, 10, 119, 680868446, 119, 683235326, 2366880

4194304, 2, 10, 119, 683251166, 119, 683314886, 63720

4194304, 3, 10, 119, 683321006, 119, 683380886, 59880

4194304, 4, 10, 119, 683388326, 119, 683445086, 56760

4194304, 5, 10, 119, 683452046, 119, 683507726, 55680

4194304, 6, 10, 119, 683513966, 119, 683566406, 52440

4194304, 7, 10, 119, 683573006, 119, 683624366, 51360

4194304, 8, 10, 119, 683630486, 119, 683684486, 54000

4194304, 9, 10, 119, 683691326, 119, 683745086, 53760

8388608, 0, 10, 119, 683751566, 119, 688888647, 5137081

8388608, 1, 10, 119, 688919367, 119, 691322367, 2403000

8388608, 2, 10, 119, 691341807, 119, 691468287, 126480

8388608, 3, 10, 119, 691476447, 119, 691601847, 125400

8388608, 4, 10, 119, 691610727, 119, 691734567, 123840

8388608, 5, 10, 119, 691741647, 119, 691862967, 121320

8388608, 6, 10, 119, 691870767, 119, 691991727, 120960

8388608, 7, 10, 119, 691998927, 119, 692149887, 150960

8388608, 8, 10, 119, 692162247, 119, 692284047, 121800

8388608, 9, 10, 119, 692291247, 119, 692412927, 121680

16777216, 0, 10, 119, 692420607, 119, 698691087, 6270480

16777216, 1, 10, 119, 698722647, 119, 701908287, 3185640

16777216, 2, 10, 119, 701931567, 119, 702532047, 600480

16777216, 3, 10, 119, 702540447, 119, 703520247, 979800

16777216, 4, 10, 119, 703528647, 119, 703806927, 278280

16777216, 5, 10, 119, 703814127, 119, 704105007, 290880

16777216, 6, 10, 119, 704115087, 119, 704399367, 284280

16777216, 7, 10, 119, 704406807, 119, 704664927, 258120

16777216, 8, 10, 119, 704672247, 119, 704933967, 261720

16777216, 9, 10, 119, 704942007, 119, 705197367, 255360

33554432, 0, 10, 119, 705206007, 119, 718937368, 13731361

33554432, 1, 10, 119, 719255848, 119, 732041368, 12785520

33554432, 2, 10, 119, 732076408, 119, 745467089, 13390681

33554432, 3, 10, 119, 745498289, 119, 758549009, 13050720

33554432, 4, 10, 119, 758579489, 119, 772443090, 13863601

33554432, 5, 10, 119, 772473930, 119, 784075530, 11601600

33554432, 6, 10, 119, 784103850, 119, 799029931, 14926081

33554432, 7, 10, 119, 799061731, 119, 812426132, 13364401

33554432, 8, 10, 119, 812463812, 119, 825670772, 13206960

33554432, 9, 10, 119, 825697652, 119, 838968933, 13271281

50331648, 0, 10, 119, 838995813, 119, 858921573, 19925760

50331648, 1, 10, 119, 858953973, 119, 879499534, 20545561

50331648, 2, 10, 119, 879529774, 119, 899781215, 20251441

50331648, 3, 10, 119, 899813015, 119, 917663136, 17850121

50331648, 4, 10, 119, 917694096, 119, 939956617, 22262521

50331648, 5, 10, 119, 939993217, 119, 958662937, 18669720

50331648, 6, 10, 119, 958694017, 119, 978534818, 19840801

50331648, 7, 10, 119, 978571298, 119, 998353539, 19782241

50331648, 8, 10, 119, 998388579, 120, 17454540, 19065961

50331648, 9, 10, 120, 17488380, 120, 38087221, 20598841

67108864, 0, 10, 120, 38126341, 120, 63385262, 25258921

67108864, 1, 10, 120, 63419462, 120, 90176103, 26756641

67108864, 2, 10, 120, 90215103, 120, 114481744, 24266641

67108864, 3, 10, 120, 114515224, 120, 141748265, 27233041

67108864, 4, 10, 120, 141783305, 120, 166453986, 24670681

67108864, 5, 10, 120, 166494426, 120, 193258027, 26763601

67108864, 6, 10, 120, 193285987, 120, 217231868, 23945881

67108864, 7, 10, 120, 217264868, 120, 243662349, 26397481

67108864, 8, 10, 120, 243694629, 120, 267378070, 23683441

67108864, 9, 10, 120, 267408310, 120, 293071151, 25662841

-- finished.

1. Result

**Table 5‑58: Result (Type4)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Array size**  **(MB)** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **2M** | 1.63740 | 1.34880 | 0.01728 | 0.00996 | 0.00948 | 0.00960 | 0.00972 | 0.00948 | 0.00936 | 0.00948 |
| **4M** | 2.22336 | 1.29840 | 0.05184 | 0.04308 | 0.04248 | 0.04440 | 0.04500 | 0.04380 | 0.04440 | 0.04380 |
| **8M** | 3.50856 | 1.97088 | 0.13080 | 0.12852 | 0.12360 | 0.12120 | 0.12156 | 0.12120 | 0.12180 | 0.12216 |
| **16M** | 6.16272 | 3.23760 | 0.25104 | 0.24816 | 0.24696 | 0.24960 | 0.24876 | 0.24660 | 0.24828 | 0.24912 |
| **32M** | 11.08860 | 10.80300 | 10.97712 | 10.92012 | 10.93188 | 10.86540 | 11.00568 | 10.93836 | 10.89228 | 10.86912 |
| **48M** | 16.34928 | 15.21552 | 15.54264 | 15.69276 | 16.03968 | 15.81336 | 15.92712 | 16.00848 | 15.96396 | 15.99540 |
| **64M** | 21.41364 | 19.84644 | 20.74308 | 20.96724 | 20.87376 | 20.91876 | 20.95980 | 20.31012 | 20.85504 | 20.67840 |

**Table 5‑59: Result (Type2)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Array size**  **(MB)** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **2M** | 1.72224 | 1.18176 | 0.01440 | 0.00984 | 0.00960 | 0.00984 | 0.00972 | 0.00960 | 0.00984 | 0.00960 |
| **4M** | 2.28300 | 1.27488 | 0.05088 | 0.04308 | 0.04260 | 0.05064 | 0.04416 | 0.04428 | 0.04248 | 0.04272 |
| **8M** | 3.53136 | 2.00724 | 0.12324 | 0.12084 | 0.13680 | 0.12480 | 0.11880 | 0.12000 | 0.11976 | 0.11964 |
| **16M** | 6.00696 | 3.31860 | 0.24864 | 0.26520 | 0.24528 | 0.24456 | 0.25056 | 0.25344 | 0.25260 | 0.26844 |
| **32M** | 11.28720 | 10.94304 | 10.96320 | 10.90296 | 10.72728 | 10.85376 | 11.14068 | 11.02704 | 10.81728 | 11.05776 |
| **48M** | 15.91368 | 15.74028 | 15.86424 | 15.99348 | 15.88404 | 15.86100 | 15.63744 | 16.13556 | 15.80496 | 15.93732 |
| **64M** | 21.41364 | 19.68180 | 20.44608 | 20.51640 | 20.91144 | 20.47980 | 20.67780 | 21.11076 | 20.79684 | 20.89596 |

Table 5‑60: Result

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Array size**  **(MB)** | **Virtualized Linux (Type4) [ns]** | | | **Native Linux (Type2) [ns]** | | |
| **Ave.** | **Min.** | **Max.** | **Ave.** | **Min.** | **Max.** |
| **2M** | 1.63740 | 0.00936 | 0.30706 | 1.72224 | 0.0096 | 0.298644 |
| **4M** | 2.22336 | 0.04248 | 0.38806 | 2.283 | 0.04248 | 0.391872 |
| **8M** | 3.50856 | 0.12120 | 0.64703 | 3.53136 | 0.1188 | 0.652248 |
| **16M** | 6.16272 | 0.24660 | 1.13888 | 6.006961 | 0.24456 | 1.1354281 |
| **32M** | 11.08860 | 10.80300 | 10.92916 | 11.2872 | 10.72728 | 10.9720204 |
| **48M** | 16.34928 | 15.21552 | 15.85482 | 16.13556 | 15.637441 | 15.8772007 |
| **64M** | 21.41364 | 19.84644 | 20.75663 | 21.413641 | 19.6818 | 20.6930528 |

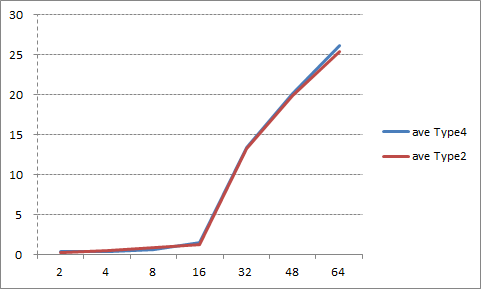


Figure 5‑16: Average Result Virtualized Linux and Native Linux (Type4 and Type2)

1. Consideration

This test result is observed quite similar to the result of 5.11.4, random write performance. As this memory allocation benchmark program performs 32-bit data write per every 4kB allocation size to force the actual allocation of the memory page, this data write is estimated to determine this memory allocation/deallocation performance benchmark.

### Read Cached/Uncached memory performance

1. Description

Measure the performance to read the cached/uncached memory on virtualized Linux and native Linux.

Measurement tool is lmbench.

1. Precondition

* Measure on virtualized Linux and native Linux (Type4 and Type2)

\*Both types stop the function of Linux App

* Use the result of Sequential Read for the smallest test size as cached performance, and the result of Random Read for the largest size as uncached performance.
* Compare the performance between virtualized Linux and native Linux. The performance results should be near to native OS implementations results.
* Verified 10 times and use the average as the result value.

1. How to measure
2. Cached performance is taken from the minimum size (0.00049MB) result of 5.6.1
3. Uncached performance is taken from the maximum size (64.000MB) result of 5.6.3

As the uncached memory access is similar to the cache miss operation, the uncached memory access performance can be substituted by the cache miss performance. The typical cached performance is the smallest size of the sequential read test (5.1.1) which the test size is enough smaller than L1 cache size (32kB), and the typical uncached performance is the largest size of the random read test (5.1.3) wich is the nearest one of the true uncache performance.

1. Result

Table 5‑61: Cached Result

|  |  |
| --- | --- |
| **Virtualized Linux**  **(Type4) [ns]** | **Native Linux**  **(Type2) [ns]** |
| 2.67 | 2.67 |

Table 5‑62: UnCached Result

|  |  |
| --- | --- |
| **Virtualized Linux**  **(Type4) [ns]** | **Native Linux**  **(Type2) [ns]** |
| 260.65 | 227.99 |

1. Consideration

This result is expected. As the uncached memory access performance is taken from the result of random access test, it includes the effect of TLB miss. See 5.11.3 for detail.















### RTOS performance

### INTEGRITY OS Performance

All measurements below using api\_measurements.gpj in release T14.0 and are in nanoseconds.

#### Deadlock break time

1. Description

Out of scope.

1. Precondition
2. How to measure

1. Result

1. Consideration

#### Semaphore processing

1. Description

1. Precondition

GHS special environment is needed. Our PoC doesn't include it.

1. How to measure

GHS special environment is needed. Our PoC doesn't include it.

1. Result

Table 5‑67: Result (nano second)

|  |  |
| --- | --- |
| TryToObtainSemaphore on available Semaphore | 321.332 |
| TryToObtainSemaphore on unavailable Semaphore | 324.627 |
| TimedWaitForSemaphore on available Semaphore | 1136.588 |
| ReleaseSemaphore no waiter | 310.069 |
| TryToClearSemaphore | 310.685 |
| GetSemaphoreValue | 312.120 |
| WaitForLocalMutex on available LocalMutex | 7.336 |
| ReleaseLocalMutex | 6.002 |
| TryToObtainLocalMutex on available LocalMutex | 6.002 |
| TryToObtainLocalMutex on unavailable LocalMutex | 6.002 |
| TryToObtainSemaphore on available Semaphore | 321.332 |

1. Consideration

This result is expected.

#### Message Passing

1. Description

1. Precondition

GHS special environment is needed. Our PoC doesn't include it.

1. How to measure

GHS special environment is needed. Our PoC doesn't include it.

1. Result

Table 5‑68: Result (nano second)

|  |  |
| --- | --- |
| 0-byte message SynchronousSend | 440.850 |
| SynchronousReceive on Object with pending message | 322.660 |
| Simple AllocateMessageQueueBuffer | 67.586 |
| Simple SendOnMessageQueue | 64.801 |
| Simple GetMessageCountForMessageQueue | 60.468 |
| Simple TimedReceiveOnMessageQueue | 65.464 |
| Simple ReceiveOnMessageQueue | 68.113 |
| Simple FreeMessageQueueBuffer | 21.527 |

1. Consideration

This result is expected.

#### Context Switching

1. Description

1. Precondition

GHS special environment is needed. Our PoC doesn't include it.

1. How to measure

GHS special environment is needed. Our PoC doesn't include it.

1. Result

Table 5‑69: Result (nano second)

|  |  |
| --- | --- |
| context-switch overhead | 680.429 |

1. Consideration

This result is expected.

#### Interrupt Latency

1. Description

1. Precondition

GHS special environment is needed. Our PoC doesn't include it.

1. How to measure

GHS special environment is needed. Our PoC doesn't include it.

1. Result

Table 5‑70: Result (nano second)

|  |  |
| --- | --- |
| Timer interrupt latency | 360.121 |

1. Consideration

This result is expected.























## Memory usage

### Check the memory usage

1. Description

Measure the memory usage of Multivisor of Center Information and Instrument Cluster / Head-up display on virtualization PoC.

Measurement tool is top command for Linux, Multi Debugger for INTEGRITY.

1. Precondition

* Measure on virtualization PoC (Type1)
* Use a tool including in Multi Debugger.

1. How to measure

Calculate a use memory size by subtracting unused memory from a total memory size (3,968 Mbyte).

Estimated memory usage = Total memory - free memory [Virtualized Linux, Memory Pool, heap]

　Note that this value is not exactly because we have omitted the small pieces of memory.

* Measurement of the unused memory size in MemoryPool

1. Select [Target] – [Connect] from Menu bar on MULTI.
2. Select “Dynamic Download/INDRT Connection (rtserv2) for Device Tree” and press “Connect” button.
3. Select “Run mode target”
4. Run the following command on “Trg” tab.

INDRT2>ct

1. Run the following command on “Trg” tab.

INDRT2>lt

After finishing a command, you will see the log like below. Check the result of the memory size (the right side) of the line of "Type1\_kernel".

Type1\_kernel 0x0000000000001000/0x000000000cb24000

0xffffffa6c571e000 exited 127 0x0000000000000630/0x0000000000008000 0.17% Initial

0xffffffa6c5720000 running 0 0x0000000000000020/0x0000000000000400 41.14% Idle0

0xffffffa6c5722000 running 0 0x0000000000000020/0x0000000000000400 61.42% Idle1

0xffffffa6c5724000 running 0 0x0000000000000020/0x0000000000000400 63.93% Idle2

Similarly, we choice the large value of result of "multivisor\_loader", "pvrserver\_as0", "devtree\_generic\_server\_module" and "Sakura".

1. Run the step 5 process 10 seconds after the result is displayed.

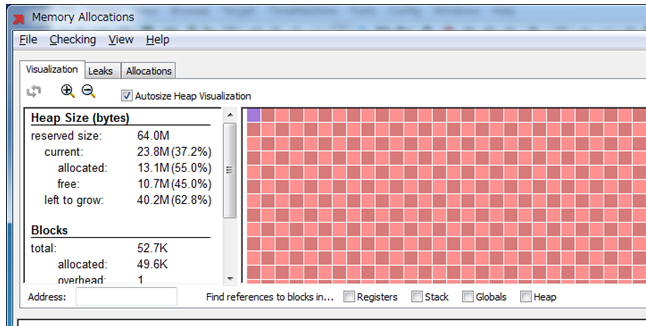
Repeat this for 10 minutes.

* Measurement of the unused memory size in Heap Memory

1. Select [Target] – [Connect] from Menu bar of MULTI
2. Select “Dynamic Download/INDRT Connection (rtserv2) for Device Tree” and press “Connect” button.
3. Expand a tree of "INTEGRITY SMP Application" in "Target View", and choose "Core 1, Cortex-A57/Type1\_mono".
4. Select [Debug] – [Halt on Selected Items] from Menu bar of MULTI
5. Run the following command on “Cmd” tab.

MULTI> heapview

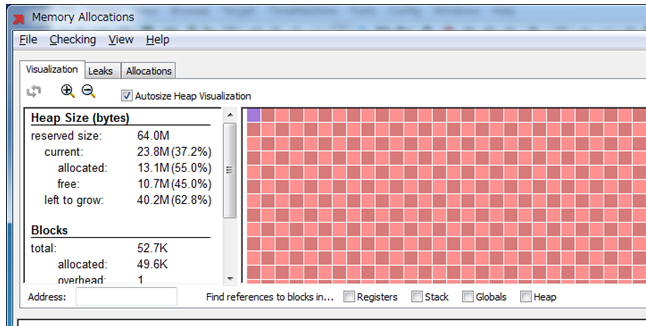
After finishing a command, you will see the log like below. Check the result of the “free” value.



1. Expand a tree of "Run mode target" in "Target View", and find “devtree\_generic\_server\_module”
2. Expand a tree of “devtree\_generic\_server\_module”, and choose “Initial”
3. Select [Debug] – [Halt on Selected Items] from Menu bar of MULTI
4. Run the following command on “Cmd” tab.

MULTI> heapview

After finishing a command, you will see the log like below. Check the result of the “free” value.



1. Continue, and run "step 6 - step 9" for the following targets.

“DISCOM\_sample\_virt”

“FBServer”

“INT\_Logo\_sample\_virt”

“ip46router\_devtree\_module”

“ivfsserver\_devtree\_module”

“multivisor\_loader”

“multivisor\_net\_server”

“multivisor\_vmm”

“pvrserver\_as0”

“Sakura”

1. Result







Table 5‑76: Unused memory in Center Information

|  |  |
| --- | --- |
|  | **Center Information**  **[Byte]** |
| **Ave.** | 862529536 |
| **1** | 885567488 |
| **2** | 877494272 |
| **3** | 871903232 |
| **4** | 870629376 |
| **5** | 867295232 |
| **6** | 861986816 |
| **7** | 857751552 |
| **8** | 854102016 |
| **9** | 843231232 |
| **10** | 835334144 |

Table 5‑77: Unused memory in Memory Pool

|  |  |
| --- | --- |
| **AddressSpace** | **MemoryPool [Byte]** |
| **Type1\_kernel** | 213008384 |
| **multivisor\_loader** | 209858560 |
| **pvrserver\_as0** | 165662720 |
| **devtree\_generic\_server\_module** | 22028288 |
| **Sakura** | 67108864 |
| **Total** | 677666816 |

Table 5‑78: Unused memory in Heap Memory

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Type1\_kernel**  **[Byte]** | **devtree\_generic\_server\_module**  **[Byte]** | **DISCOM\_sample\_virt**  **[Byte]** | **FBServer**  **[Byte]** | **INT\_Logo\_sample\_virt**  **[Byte]** | **ip46router\_devtree\_module**  **[Byte]** |
| **Ave.** | 35574 | 12083 | 10824 | 11674 | 10711 | 4813 |
| **1** | 34918 | 12083 | 10752 | 11674 | 10752 | 4812 |
| **2** | 35737 | 12083 | 10854 | 11674 | 10650 | 4812 |
| **3** | 35737 | 12083 | 10854 | 11674 | 10650 | 4812 |
| **4** | 35737 | 12083 | 10854 | 11674 | 10650 | 4812 |
| **5** | 35430 | 12083 | 10854 | 11674 | 10650 | 4812 |
| **6** | 34918 | 12083 | 10854 | 11674 | 10752 | 4812 |
| **7** | 35942 | 12083 | 10854 | 11674 | 10752 | 4812 |
| **8** | 35942 | 12083 | 10752 | 11674 | 10752 | 4812 |
| **9** | 35635 | 12083 | 10752 | 11674 | 10752 | 4812 |
| **10** | 35737 | 12083 | 10854 | 11674 | 10752 | 4812 |
|  | **ivfsserver\_devtree\_module**  **[Byte]** | **multivisor\_loader**  **[Byte]** | **multivisor\_net\_server**  **[Byte]** | **multivisor\_vmm**  **[Byte]** | **pvrserver\_as0**  **[Byte]** | **Sakura**  **[Byte]** |
| **Ave.** | 13537 | 10547 | 7373 | 65536 | 540201 | 13956547 |
| **1** | 16486 | 10547 | 7373 | 65536 | 600781 | 11324621 |
| **2** | 13209 | 10547 | 7373 | 65536 | 657510 | 13316915 |
| **3** | 13209 | 10547 | 7373 | 65536 | 384410 | 14365491 |
| **4** | 13209 | 10547 | 7373 | 65536 | 441242 | 14365491 |
| **5** | 13209 | 10547 | 7373 | 65536 | 441242 | 14365491 |
| **6** | 13209 | 10547 | 7373 | 65536 | 711270 | 14365491 |
| **7** | 13209 | 10547 | 7373 | 65536 | 512000 | 14365491 |
| **8** | 13209 | 10547 | 7373 | 65536 | 541082 | 14365491 |
| **9** | 13209 | 10547 | 7373 | 65536 | 556339 | 14365491 |
| **10** | 13209 | 10547 | 7373 | 65536 | 556134 | 14365491 |

Table 5‑79: Result

|  |  |
| --- | --- |
| Estimated memory usage | 4160749568 – (862529536 + 677666816 + 14679419 ) = 2605873797 (Byte)  =2485.16(MByte) |

1. Consideration

This result is expected

## Memory Utilization of Each Module

### Memory utilization in IVI (Center Information)

1. Description

Measure RAM utilization in Center Information application on virtualization PoC using top command on virtualization PoC.

1. Precondition

* Measure on virtualized Linux on virtualization PoC (Type1)
* Use top command.
* Verified 10 times and use the average as the result value.

1. How to measure
2. Refers to 5.19.1.
3. Result

Table 5‑80: Result

|  |  |
| --- | --- |
| **Virtualization PoC (Type1)** | **Value**  **[Kbyte]** |
| **Center Information** | 1340790.00 |

1. Consideration

This result come from the average of Table 5-73. This result is expected.

### Memory utilization in meter (Instrument Cluster)

1. Description

Measure RAM utilization in Instrument Cluster application on virtualization PoC using Command prompt.

The following applications are measurement target. Refer to Chapter 4 for details.

- Meter Cluster app

- OGLES31

- Display driver

1. Precondition

* Measure on INTEGRITY on virtualization PoC (Type1)
* Use a command on Command prompt.

1. How to measure

Calculate a used memory size by measurement for several kinds of memory sizes that each application uses, and adding each.

Memory utilization = MemoryPool + heap memory + ROM/RAM size + Memory Region.

* Measurement of the used memory size in MemoryPool

Measurement method refers to “Measurement of the unused memory size in MemoryPool” of 5.19.1 .

Excerpt from results of 5.19.1. Check the following results of the memory size (the left side) of the line of "Sakura".

Sakura 0x00000000000cb000/0x0000000004000000

0xffffffa6c57f0000 running 127 0x00000000000187b0/0x0000000000200000 31.39% Initial

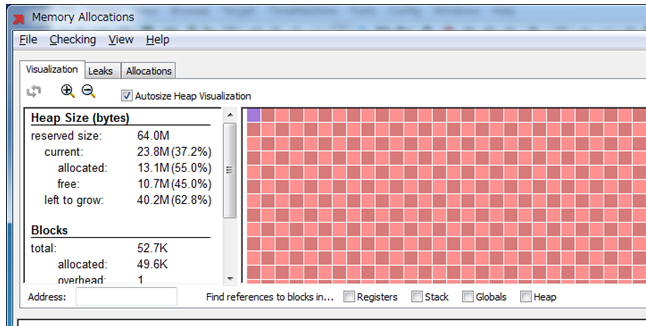
0xffffffa00fae3000 pending 128 0x0000000000000168/0x0000000000003cb8 0.00% PosixServer

Similarly, check the result of "pvrserver\_as0" and "FBServer".

* Measurement of the used memory size in Heap Memory

Measurement method refers to “Measurement of the unused memory size in MemoryPool” of 5.19.1 .

Excerpt from results of 5.19.1. Check the following results of the memory size (the left side) of the line of "Sakura".



* Measurement of the used memory size in ROM/RAM size

1. Run the following command.(measurement ROM/RAM size of Sakura)

C:\>C:\ghs\comp\_201516\gsize.exe -ram -rom C:\Sakura-h3

After finishing a command, you will see the log like below.

Red square is results.

C:\ghs\comp\_201516\gsize.exe -ram -rom C:\Sakura-h3

RAM\_Size: 1828492

ROM\_Size: 19707803

1. Run the following command. (measurement ROM/RAM size of pvrserver\_as0)

C:\>C:\ghs\comp\_201516\gsize.exe -ram -rom C:\ghs\int1144\bin\devtree-arm64\pvrserver\_as0

After finishing a command, you will see the log like below.

Red square is results.

C:\>C:\ghs\comp\_201516\gsize.exe -ram -rom C:\pvrserver\_as0

RAM\_Size: 109684

ROM\_Size: 784716

1. Run the following command. (measurement ROM/RAM size of FBServer)

C:\>C:\ghs\comp\_201516\gsize.exe -ram -rom C:\FBServer

After finishing a command, you will see the log like below.

Red square is results.

C:\>C:\ghs\comp\_201516\gsize.exe -ram -rom C:\FBServer

RAM\_Size: 23148

ROM\_Size: 122484

* Measurement of the Memory Region size.

The memory region of sakura use maximum following size.

Memory Region Size : 147540214 (Byte)

1. Result

* Instrument Cluster

- Sakura (Meter Cluster app)

- pvrserver\_as0 (OGLES31)

- FBServer (Display driver)

Table 5‑81: Used memory Pool

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Sakura**  **(Byte)** | **pvrserver\_as0**  **(Byte)** | **FBServer**  **(Byte)** |
| **Ave.** | 895385.6 | 40686387.2 | 0 |
| **1** | 872448 | 40693760 | 0 |
| **2** | 913408 | 40701952 | 0 |
| **3** | 913408 | 40701952 | 0 |
| **4** | 913408 | 40685568 | 0 |
| **5** | 880640 | 40677376 | 0 |
| **6** | 880640 | 40677376 | 0 |
| **7** | 880640 | 40677376 | 0 |
| **8** | 921600 | 40685568 | 0 |
| **9** | 921600 | 40685568 | 0 |
| **10** | 856064 | 40677376 | 0 |

Table 5‑82: Used heap memory

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Sakura**  **(Byte)** | **pvrserver\_as0**  **(Byte)** | **FBServer**  **(Byte)** |
| **Ave.** | 13715374 | 12457083 | 568 |
| **1** | 13631488 | 12373197 | 568 |
| **2** | 13631488 | 12478054 | 568 |
| **3** | 13736346 | 12478054 | 568 |
| **4** | 13736346 | 12478054 | 568 |
| **5** | 13736346 | 12478054 | 568 |
| **6** | 13736346 | 12373197 | 568 |
| **7** | 13736346 | 12478054 | 568 |
| **8** | 13736346 | 12478054 | 568 |
| **9** | 13736346 | 12478054 | 568 |
| **10** | 13736346 | 12478054 | 568 |

Table 5‑83: Result

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **AddressSpace** | **MemoryPool**  **(Byte)** | **Heap Memory**  **(Byte)** | **ROM size**  **(Byte)** | | **RAM size**  **(Byte)** | | **Memory Region**  **(Byte)** |
| **Sakura** | 895385.6 | 13715374 | 19707803 | | 1828492 | | 147073269.8 |
| **pvrserver\_as0** | 40686387.2 | 12457083 | 784716 | | 109684 | | 466944 |
| **FBServer** | 0 | 568 | 122484 | | 23148 | | 0 |
| **Total** | 41581772.8 | 26173025 | 20615003 | | 1961324 | | 147540213.8 |
| **Used memory Size** | | 237628037(Byte)  = 226.62 (MByte) | |

1. Consideration

This results is expected.

### Memory utilization in HUD (Head-up display)

1. Description

Measure RAM utilization in Head-up display application on virtualization PoC using Command prompt.

The following applications are measurement target. Refer to Chapter 4 for details.

- Telltail app

- Display driver

1. Precondition

* Measure on INTEGRITY on virtualization PoC (Type1)
* Use a command on Command prompt.

1. How to measure

Calculate a used memory size by measurement for several kinds of memory sizes that each application uses, and adding each.

Memory utilization = MemoryPool + heap memory + ROM/RAM size + Memory Region.

* Measurement of the used memory size in MemoryPool

Measurement method refers to “Measurement of the unused memory size in MemoryPool” of 5.19.1 .

Excerpt from results of 5.19.1. Check the following results of the memory size (the left side) of the line of "Telltale".

DISCOM\_sample\_virt 0x0000000000005000/0x0000000000080000

0xffffffa6c57f8000 pending 127 0x00000000000009d0/0x0000000000008000 1.61% Initial

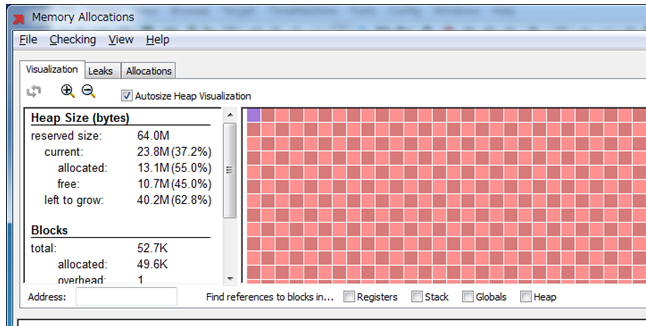
0xffffffa00ff26000 pending 254 0x0000000000000280/0x0000000000000dd0 0.00% OSAAgent

Similarly, check the result of "pvrserver\_as0" and "FBServer".

* Measurement of the used memory size in Heap Memory

Measurement method refers to “Measurement of the unused memory size in MemoryPool” of 5.19.1 .

Excerpt from results of 5.19.1. Check the following results of the memory size (the left side) of the line of "DISCOM\_sample\_virt".



* Measurement of the used memory size in ROM/RAM size

1. Run the following command.(measurement ROM/RAM size of Sakura)

C:\ghs\comp\_201516\gsize.exe -ram -rom

After finishing a command, you will see the log like below.

Red square is results.

C:\ghs\int1144\bin\devtree-arm64\DISCOM\_sample\_virt

RAM\_Size: 285296

ROM\_Size: 1150472

1. Run the following command. (measurement ROM/RAM size of FBServer)

C:\>C:\ghs\comp\_201516\gsize.exe -ram -rom C:\FBServer

After finishing a command, you will see the log like below.

Red square is results.

C:\>C:\ghs\comp\_201516\gsize.exe -ram -rom C:\FBServer

RAM\_Size: 23148

ROM\_Size: 122484

* Measurement of the used memory size in ROM/RAM size

The memory region of Telltale use maximum following size.

Memory Region Size : 2334720 (Byte)

1. Result

* Instrument Cluster

- DISCOM\_sample\_virt (Telltale)

- FBServer (Display driver)

Table 5‑84: Used memory Pool

|  |  |  |
| --- | --- | --- |
|  | **Telltale**  **(Byte)** | **FBServer**  **(Byte)** |
| **Ave.** | 20480 | 0 |
| **1** | 20480 | 0 |
| **2** | 20480 | 0 |
| **3** | 20480 | 0 |
| **4** | 20480 | 0 |
| **5** | 20480 | 0 |
| **6** | 20480 | 0 |
| **7** | 20480 | 0 |
| **8** | 20480 | 0 |
| **9** | 20480 | 0 |
| **10** | 20480 | 0 |

Table 5‑85: Used heap memory

|  |  |  |
| --- | --- | --- |
|  | **Sakura**  **(Byte)** | **FBServer**  **(Byte)** |
| **Ave.** | 1464 | 568 |
| **1** | 1536 | 568 |
| **2** | 1434 | 568 |
| **3** | 1434 | 568 |
| **4** | 1434 | 568 |
| **5** | 1434 | 568 |
| **6** | 1434 | 568 |
| **7** | 1434 | 568 |
| **8** | 1536 | 568 |
| **9** | 1536 | 568 |
| **10** | 1434 | 568 |

Table 5‑86: Result

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **AddressSpace** | **MemoryPool**  **(Byte)** | **Heap Memory**  **(Byte)** | **ROM size**  **(Byte)** | | **RAM size**  **(Byte)** | | **Memory Region**  **(Byte)** |
| **DISCOM\_sample\_vir** | 20480 | 1464 | 1150472 | | 285296 | | 2334720 |
| **FBServer** | 0 | 568 | 122484 | | 23148 | | 0 |
| **Total** | 20480 | 2036 | 1272956 | | 308444 | | 2334720 |
| **Used memory Size** | | 3938632(Byte)  = 3.76 (MByte) | |



1. Consideration

This results is expected.