Optimizing I/O Virtualization & VM Memory Management for Tablet Devices

October 14th, 2014

Bokdeuk Jeong, Jaeyong Yoo, Sung-Min Lee
Software R&D Center, Samsung Electronics
Vasily Leonenko
Samsung R&D Institute Russia, Samsung Electronics

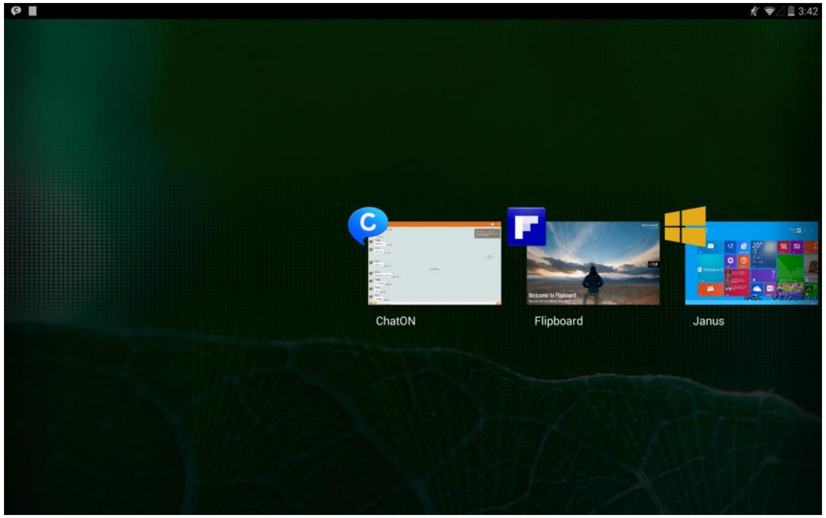
QEMU/KVM

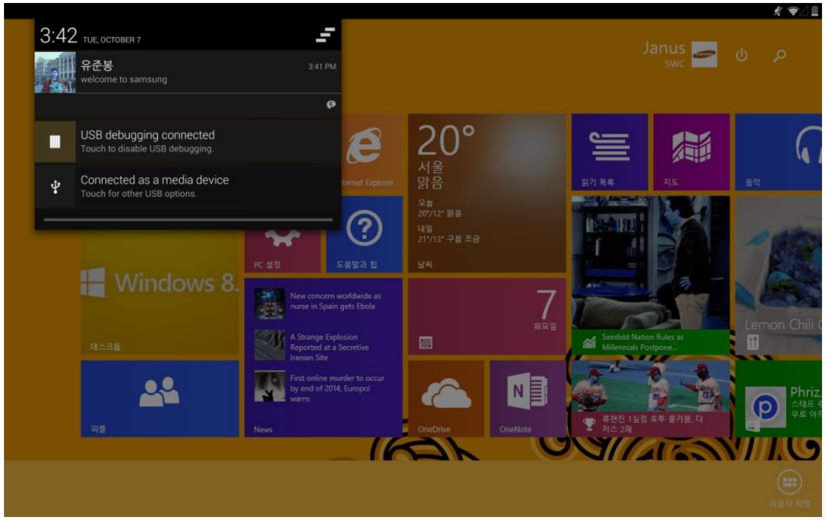
RUNNING WINDOWS ON ANDROID

Running Another OS on Android

- Windows 8.1 on Android KitKat w/ Atom Tablet

Looks like this ➔





3:42 TUE, OCTOBER 7

유준봉 3:41 PM
welcome to samsung

USB debugging connected
Touch to disable USB debugging.

Connected as a media device
Touch for other USB options.

HW spec:
Baytrail (Atom Z3775, 1.46 GHz)
3GB RAM
64GB eMMC 5.0

VM:
4 VCPUs, 1GB RAM
16GB used (30GB disk image)
Windows 8.1 32bit

Running Another OS on Android

- How To Run Windows on Android with KVM/QEMU
  - Limbo-Android
    - Runs QEMU on Android
  - Intel's Talk at KVM Forum 2013
    - Enabled Limbo w/ KVM support
    - Added missing system calls & POSIX functions
  - Samsung did
    - Rebased Limbo to QEMU 1.7.1
    - Used 32-bit Android Kernel with PAE
      for Windows to use the NX bit
    - VM w/ more than 1GB RAM support on 32-bit Android kernel

- Samsung also added
  - Multitouch
    - USB multitouch support
  - Bluetooth
    - Bluetooth pass-through
  - WIFI Access
    - Wi-Fi access via virtual Ethernet
  - Battery charge status sync
    - via ACPI
  - Audio
    - Interface with OpenSLES
  - Graphic Virtualization based on API-forwarding

Finding Another Missing Pieces

- **Running Windows 8.1 VM on Android as an App**
  - **An app should start instantly.**
    - "5 seconds is the max time most mobile user will wait for a website or application to load. 74% will leave a mobile website if it doesn't load within 5 seconds, and 50% will exit an app." (infographic)

    ➔ **Start VM with a snapshot**
  - **Android kills an app in background with the heaviest memory usage first when it is low on memory**
    - By Low Memory Killer in Kernel
    - By Activity Manager Service

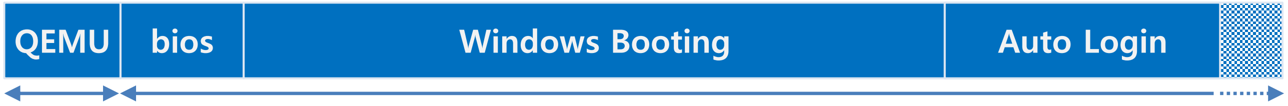
    ➔ **Utilize automatic VM memory balloon & suspend/resume**
  - **Virtualized I/O devices in QEMU should interface with Android world**
    - HIDs, Audio, Bluetooth, Battery Charging Status, Display and etc.

QEMU/KVM

STARTING UP VM WITHIN 5 SECONDS

Fast VM Startup

- Displaying Metro UI after Boot



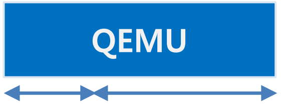
48.3 ~ 50 Seconds

3.3 seconds
for initialization

45 seconds
- with no change from the previous VM run

+ ∝
- If changes in VM occurred
  during the last run
- If disk image is not clean

- VM Startup from Snapshot* w/ Existing QEMU

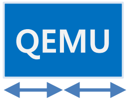


10.4 seconds

3.0 seconds
for initialization

7.4 seconds for loading VM snapshot

- VM Startup from Snapshot* w/ QEMU Modification



5.4 Seconds

1.2 seconds
for initialization

4.2 seconds for loading VM snapshot

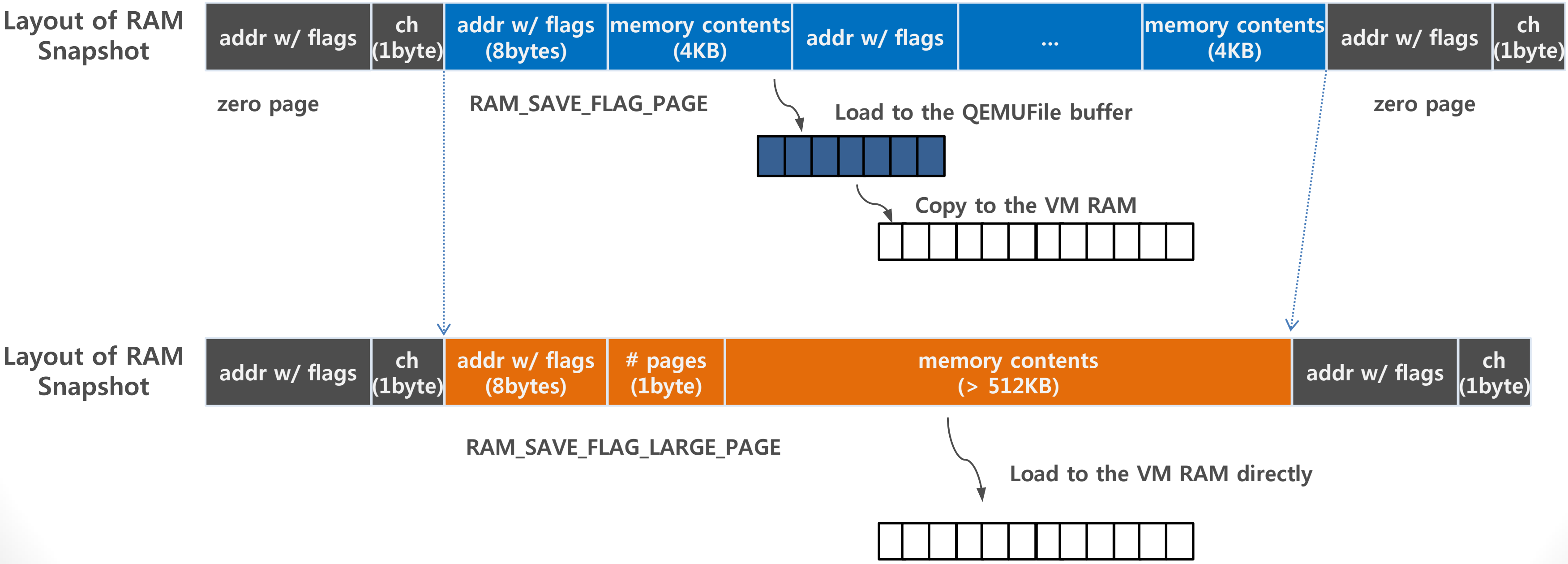
* Snapshot size: 574MB (taken at the Metro UI screen)

Fast VM Startup

- **Modification on Saving and Loading Snapshots (1/2)**
  - Separate file for VM State only
    - Reduces time for L1/L2 table lookup in qcow2
    - Reduces time for meta data lookup of a qcow2 file (> 15GB) in the host kernel
  - Resize QEMUFile buf from 32KB to 512KB
  - Read-ahead hint API
    - Set read-ahead for snapshot area in the qcow2 snapshot file
    - Using fadvise() with POSIX_FADV_SEQUENTIAL

Fast VM Startup

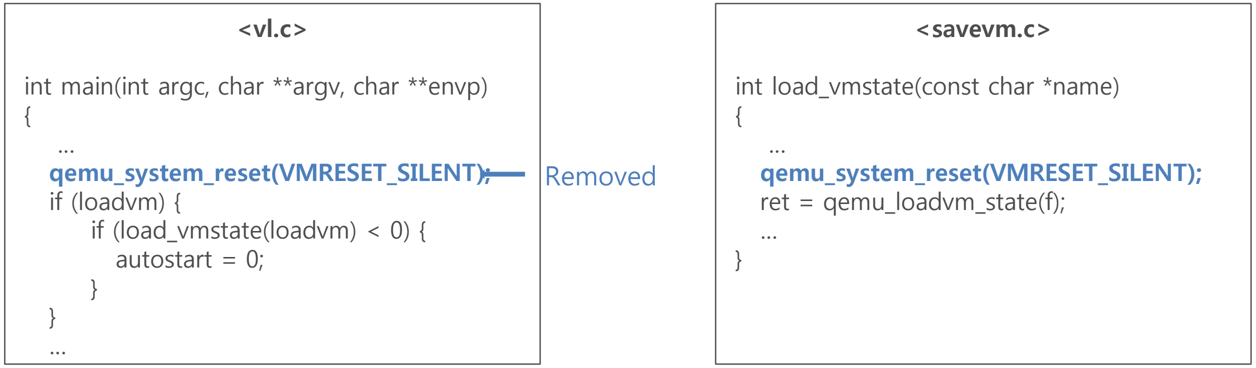
- **Modifications on Saving and Loading Snapshots (2/2)**
  - Save contiguous non-zero pages larger than 512KB together after a single header
  - Load the contiguous non-zero RAM state directly to VM RAM
    - Avoids memory copy overhead



Fast VM Startup

- **Disable unused virtual device & modules**
  - floppy disk, vmmouse
  - USB companion controllers
  - qemu monitor
  - qmp socket

- **Enable *nodefaults* option**
- **THP w/ zero-pages disabled**
- **Eliminate redundant qemu_system_reset() call**



<vl.c>

```
int main(int argc, char **argv, char **envp)
{
    ...
  qemu_system_reset(VMRESET_SILENT);
  if (loadvm) {
      if (load_vmstate(loadvm) < 0) {
          autostart = 0;
      }
  }
  ...
```

Removed

<savevm.c>

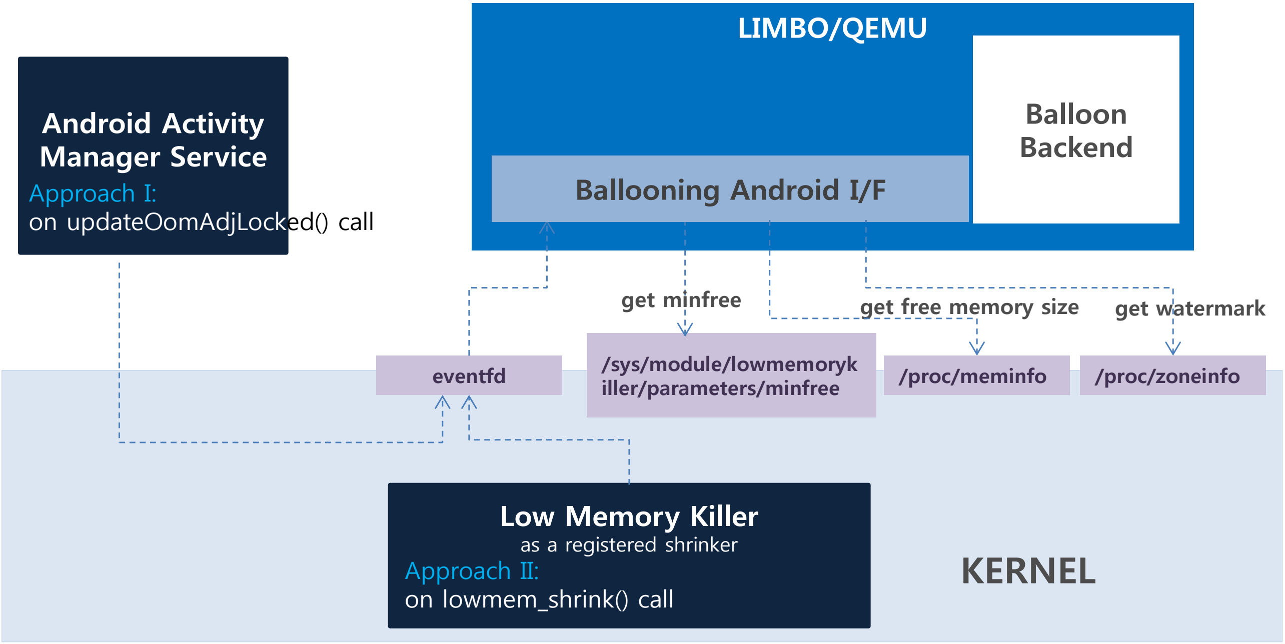
```
int load_vmstate(const char *name)
{
    ...
  qemu_system_reset(VMRESET_SILENT);
  ret = qemu_loadvm_state(f);
  ...
}
```

QEMU/KVM

BALLOONING VM MEMORY ACCORDING TO THE FOREGROUND SCREEN DISPLAY

Android-based Ballooning

AMS callback + LMK callback



- ## MinFree Table of LMK

Process type	Foreground	Visible	Perceptible, Backup	Heavy Weight, Service, Home, Previous, Service_B	Cached App Min	Cached App Max, Unknown
Adj Value	0~	1~	2 ~	4~	9~	15~
Oom Min Free	48MB	60MB	72MB	84MB	96MB	128MB

Ballooning: VM Execution at Foreground

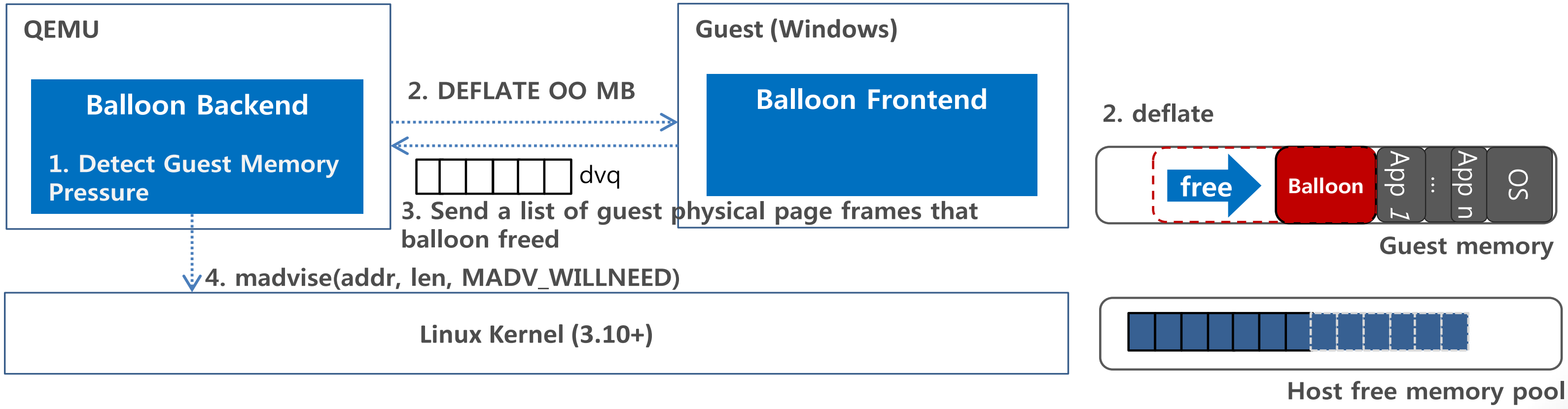
Guest has higher priority of using memory. Let VM freely use its memory

- Try to keep guest VM memory pressure small

- VM memory pressure = (Total Mem – Free Mem)/Total Mem
- VM memory pressure < FG_VM_Mem_Pressure (e.g. 75%)

- But, don't let Android sacrifice important apps for VM

- important apps: Visible apps, Perceptible apps, Services ..
- e.g.) FG_Host_Mem_Threshold = Minfree[3] (e.g. 84MB)



Ballooning: VM Execution at Background

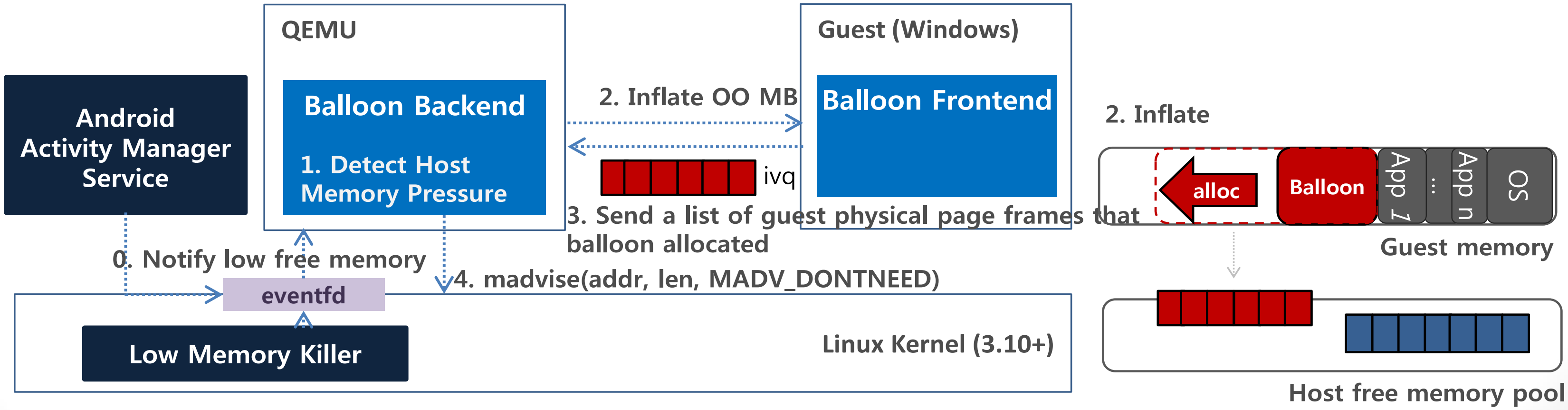
Host apps have higher priority of using memory. Yield guest memory as much as it can.

- Try to keep host free memory > BG_Host_Mem_Threshold

BG_Host_Mem_Threshold = MAX(Minfree[5],
SUM(low watermark of lowmem, low watermark of highmem))
+ margin

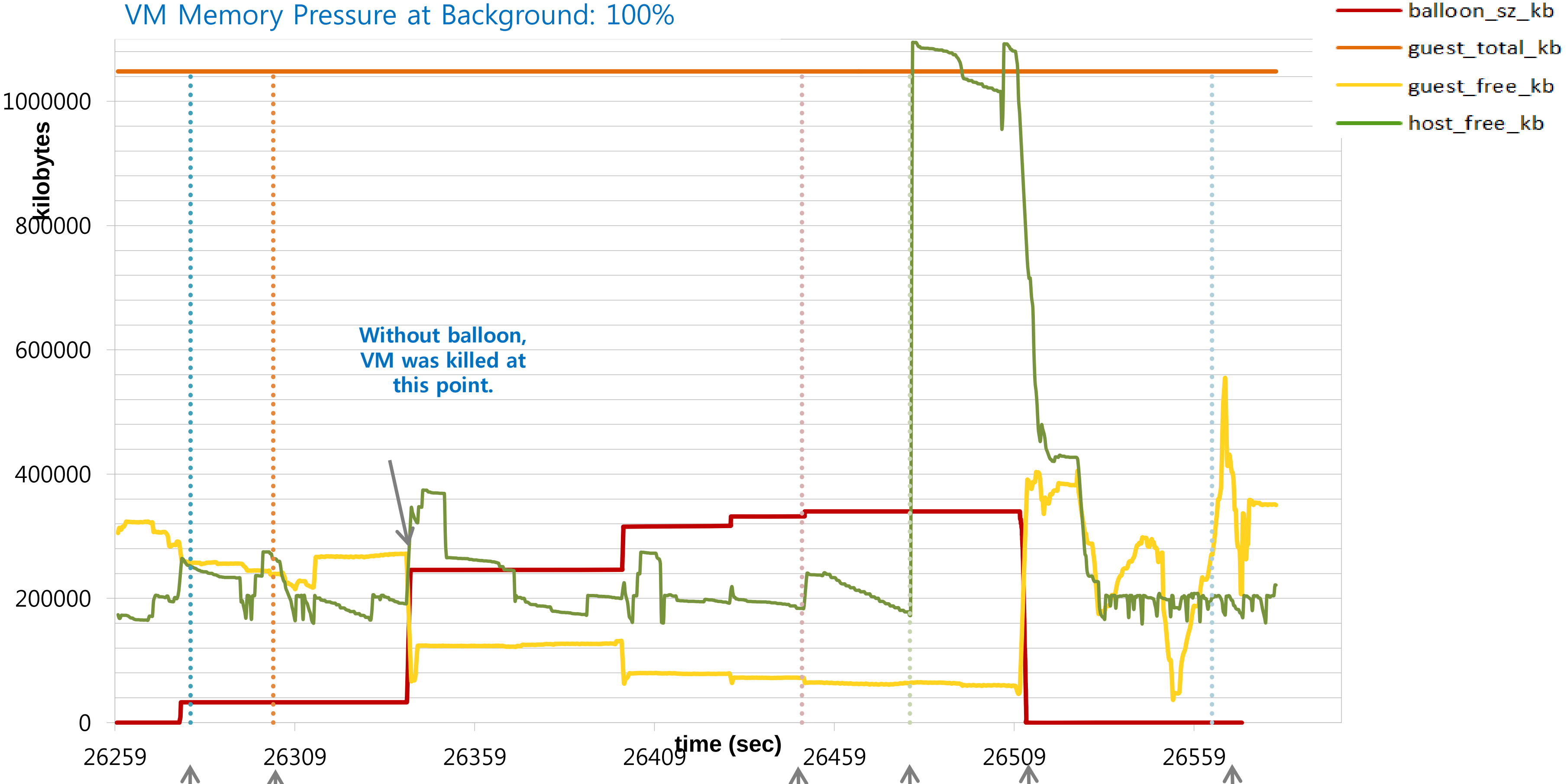
- But, don't cause severe guest page swap

VM memory pressure < BG_VM_Mem_Pressure (e.g 95%)



Ballooning: Experiment Result

Guest VM RAM: 1GB
VM Memory Pressure at Background: 100%



Without balloon, VM was killed at this point.

Switch VM -> Android

A host app starts allocating memory.

The host app has allocated 300MB

The host app has allocated 800MB

Killed the host app

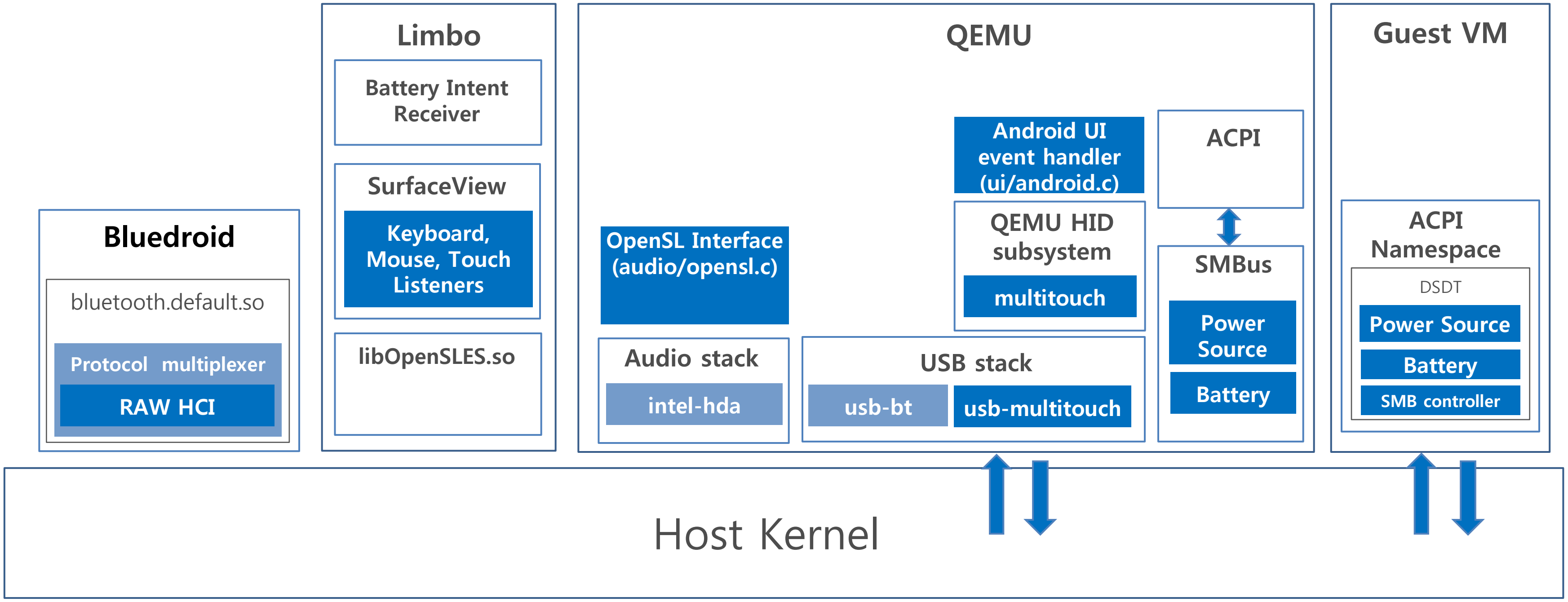
Switch Android -> VM

Killed apps in VM

QEMU/KVM

INTERFACING ANDROID WITH QEMU VIRTUAL DEVICES

I/O Devices: Newly Added Virtual Devices

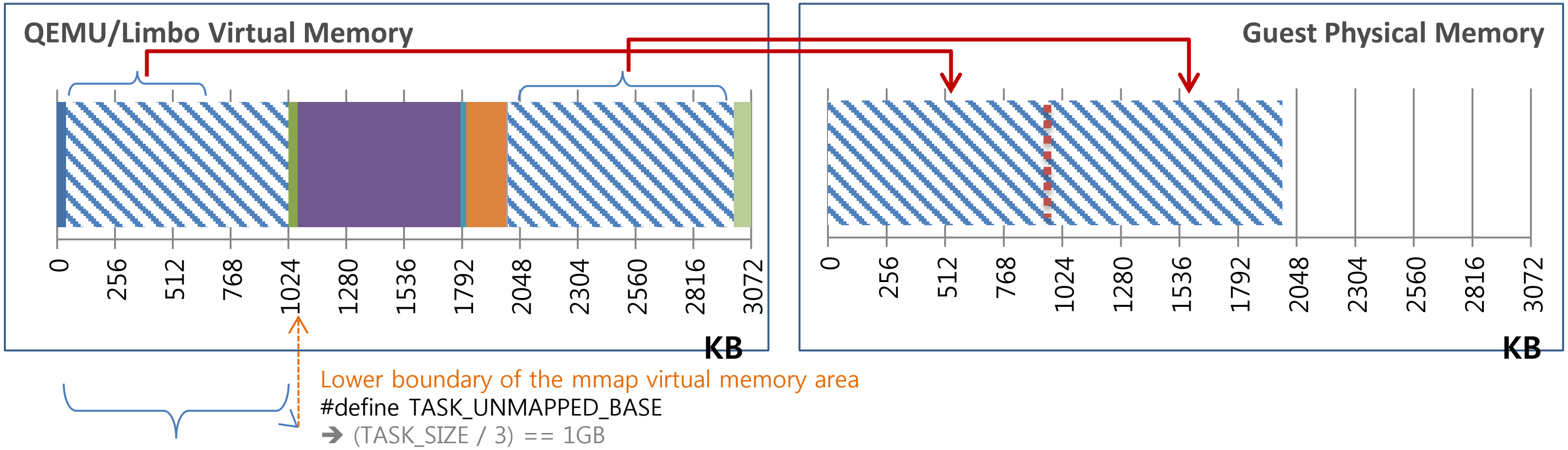


▇ newly added

I/O Devices: Features

- **HID**
  - USB Multi-touch
  - Mouse 3-buttons, hover functionalities
- **Bluetooth**
  - Bluetooth HCI Pass-through
  - Bluedroid modification to support HCI raw data
- **Sound**
  - OpenSL interface in QEMU maintains a lock-free ring buffer to pass samples to OpenSL
  - Used asynchronous queue processing maintained by OpenSL
- **ACPI**
  - Added Objects to ACPI namespace in Guest VM
    - Power Source: implements standard ACPI power source protocol
    - Battery: implements standard ACPI control method battery protocol
    - SMBus Controller: connection based on SMBus

Large VM Memory Support on 32-bit kernel



Lower boundary of the mmap virtual memory area

#define TASK_UNMAPPED_BASE

➔ (TASK_SIZE / 3) == 1GB

Unused area by Limbo (i.e. android apps)

(Linux process used to allocate this area for Text, Data, BSS segments)

Acknowledgment

- **Many thanks for individual contribution**
  - Je-Min Kim
  - Jae Min Kim
  - Sungjin Kim
  - Byung Chul So
  - Min Kang
  - Yong Hyeok Lee
  - Junbong Yu
  - Chanju Park
  - Anatoly Stepanov
  - Petr Ovchenkov
  - Sergey Vorm
  - Stanislav Andreev
  - Anton Yakovlev
  - Svetlana Khafizova

Thank you