2023 - Common Device I/F between Virt-AGL and Non-Virt AGL - Phase 2

Plan and objectives

Device	Status	Start date	First Patch release (gerrit)	Expected due date
sound	**done**	2023-07-03	week 41	~End of Oct. 2023
GPIO	**done**	2023-08-28	week 40	~End of Oct. 2023
CAN	**done**	2023-10-09	week 44	08/12/2023
console	**done**	2023-11-06	week 51	end of 2023
cloud task	ongoing			AGL AMM Feb 2024
GPU	ongoing	2023-07-03		AGL AMM Feb 2024

Weekly activity reports

This page contains weekly reports of the Virtual Open Systems activity about the "AGL Native VIRTIO phase 2" project.

week 28

Sync status slides

LF-project-status-2024-01-17.pdf

Console & Can

Follow the PRs' review process and answer to any incoming questions
The PRs can be found in the link below:
- [vhost-device-console](https://github.com/rust-vmm/vhost-device/pull/601)
- [vhost-device-can](https://github.com/rust-vmm/vhost-device/pull/602)

Benchmarks:

We have designed two device agnostic benchmark tests for measuring and comparing latency and throughput between virtio-loopback and "QEMU & vhost-user" cases.
The latency benchmark has been completed and we have already the final results by performing it on a RPI4. Lastly, this week we designed and we are at the process to finalize the implementation of throughput benchmark. Next week, out efforts will be focused to complete the throughput benchmark and start creating a presentation which includes the overall results.

GPU:

> *The work focused on investigating runtime errors of vhost-user-gpu & virtio-gpu device models on virtio-loopback-adapter. This includes sequences of*

> *virtio-gpu protocol requests of the backend (contrib/vhost-user-gpu) and the actions performed by the virtio-gpu device support on virtio-loopback-adapter.*

> *Such requests are mishandled by adapter's virtio-gpu support resulting in crash/hangup and returning invalid data to the rendering pipeline. Started an effort*
> *to rewire the virtio-gpu model emulation on a QEMU process and let virtio-loopback-adapter handle only the transport path of 'virtqueues' using virtio-loopback.*

> *Next steps include the analysis, resolution of the virtio-gpu device model integration and wiring on virtio-loopback-adapter.*

week 27

Console & Can:
One pull request (PR) for each device has been sent to rust-vmm community [github repo](https://github.com/rust-vmm/vhost-device).
The PRs can be found in the link below:
- [vhost-device-console](https://github.com/rust-vmm/vhost-device/pull/601)
- [vhost-device-can](https://github.com/rust-vmm/vhost-device/pull/602)
Next steps are to follow up with the review/comments coming from the community.

GPU:
The work focused on AGL-Yocto integration of vhost-user-gpu support for virtio-loopback, testing the performance of vhost-user-gpu and running the solution on the ARM64 reference-hardware platform.
Created a patch for the virtio-loopback-driver recipe with the necessary kernel configuration options to support the virtio-gpu driver when agl-egvirt feature is enabled. Continued the porting effort of virtio-loopback-adapter vhost-user-gpu support to the Yocto building system and resolved library linking errors on libvirglrenderer.
Next steps include the upstreaming of vhost-user-gpu support on virtio-loopback-adapter that is now in testing phase. Further analysis of the performance of virtio-loopback/virtio-gpu running the Ubigine/Valley benchmark on ARM64 and x86 host machines.

week 26

Console & Console

Finalize the code and start preparing RFCs for rust-vmm community (vhost-device crate).
Next week a pull request for each device will be sent on rust-vmm community [github repo](https://github.com/rust-vmm/vhost-device)

Sound:
This week we focused on cross-compiling issues related to vhost-device-sound. We were able to find the solution for cross-compiling it manually, with the following steps:

Requirements:

- gcc-aarch64-linux-gnu (debian package)

Build process: 1) clone latest repo: git clone https://github.com/rust-vmm/vhost-device 2) Add a new target for cross-compilation: rustup add target aarch64-unknown-linux-gnu 3) navigate into the repo dir: cd vhost-device 4) Export pkg arm64 lib and enable cross-compilation: a) `export PKG_CONFIG_PATH=/usr/aarch64-linux-gnu/lib` // The path and name might be different b) `export PKG_CONFIG_ALLOW_CROSS=1` 5) cross-compile vhost-device-sound bin: `RUSTFLAGS="-C linker=aarch64-linux-gnu-gcc" cargo build --bin vhost-device-sound --target=aarch64-unknown-linux-gnu`
Next step is now to verify the yocto configuration aiming to match the two.

GPU:
We are now able to run weston and glmark2 applications on top of virtio-loopback vhost-user-gpu using a VM as host system (for development purposes). We are now porting the same environment on the RefHW and we will come back to you next week with numbers.

week 25

gerrit reviews:
Reviews for 29545 and 29539 arrived today and are now merged. I am now rebasing my patches to submit related changes for 29539

sound:
We are working on the vhost-user-sound recipe. However, there is an issue with the rust cross-compilation of the sound device that is preventing us from building it using yocto. We are now investigating to see what's the issue and how to solve it. More info next week.

GPU:
The activity continued on testing 3D workload applications and the glmark2 benchmark. Identified an issue on surfaces shared on newly created file descriptors. Launching a new application, such as the Weston compositor or a 3D workload application, drives the vhost-user-gpu application to create a file-descriptor that is transferred to virito-loopback-adapter via unix-socket. Worked on identifying such cases and modified the adapter to retrieve the data frames from the correct file-descriptor. Collected benchmark data from the QEMU+vhost-user-gpu use case by running the glmark2 benchmark. With the virtio-loopback solution the glmark2 benchmark was not able to successfully execute the whole benchmark.
Following work includes fixes on the virtio-loopback-adapter for correct execution of applications (launching/closing multiple applications)and the execution of the glmark2 benchmark.

week 24

> *gerrit reviews:*
> *This week we pushed the console device (29545) and we updated 29539. Looking forward to get reviews so that we can proceed with sound and adapter.*
>
> *Console*
> *Review the code and prepare it for an RFC on rust-vmm community for vhost-device crate.*
> *Next step is to create a pull request for each device on vhost-device [github repo](https://github.com/rust-vmm/vhost-device)*

> *Sound:*
> *I had a sync with Yamaguchi-san this week and he mentions being stuck on the fact that AGL crosssdk that does not include cargo tool by default and by consequence he is not able to cross-compile. @Jan-Simon Moeller do you have any hints on this? I'll contact Yamaguchi-san again next week to try to solve the issue.*
>
> *In addition, this week we focused on design and implementation of benchmarks.*

> *We created a latency benchmark which measures a round trip from vhost-device-sound to virtio-sound and back.*

> *Next steps, is to run this benchmark and compare the results between the following two scenarios:*
>
> *a) Qemu - vhost-device-sound*
>
> *b) Virtio-loopback - vhost-device-sound*

week 23

gerrit reviews:
virtio-loopback-driver patch to support both qemu-x86 and qemu-arm64 was pushed and merged. Also, a new version of 29407 was prepared and merged. Lastly, we started the integration of console with 29539. The yocto part will come next.

Meta-aws
With the latest virtio-loopback-driver patch that supports both qemu-x86 and qemu-arm64, we are now in line with the execution of virtio-loopback in AWS. We will test the image once Jan-Simon completes the initial support for the AMI creation.

Console & CAN upstreaming
We started preparing a pull request for the vhost-device rust-vmm community which will be ready next week.

Sound:

A new multi-sound device guide is created for Yamaguchi-san to use. It describes all the steps required to reproduce the demo presented during OSS23

GPU:
The work continued on integration testing of the vhost-user-gpu protocol requests. In particular tested the functionality of VHOST_USER_GPU_GET_DISPLAY_INFO, VHOST_USER_GPU_GET_EDID and VHOST_USER_GPU_SCANOUT which are requested by the virtio-gpu driver in order to retrieve and setup the requirements from the display-server side.
The VHOST_USER_GPU_CURSOR_POS/CURSOR_POS_HIDE/CURSOR_UPDATE requests are implemented as dummy functions at this moment. Continued on testing the requests that are carrying the frames-payload from the contrib/vhost-user-gpu application to the virtio-loopback-adapter focused on the VHOST_USER_GPU_UPDATE/DMABUF_UPDATE,VHOST_USER_GPU_SCANOUT and VHOST_USER_GPU_DMABUF_SCANOUT. Successfully tested the acquisition and transfer of DMABUF file-descriptor resources from the vhost-user-gpu application to the adapter.
The following work includes the functional test and benchmark of the vhost-user-gpu support on virtio-loopback-adapter. The benchmark use-case will compare 3D workloads running on virito-loopback and compared with the frame rate achieved by vhost-user-gpu when running with a QEMU VM.

week 22

gerrit reviews:
virtio-loopback-driver patch to support both qemu-x86 and qemu-arm64 was pushed and merged. Also, a new version of 29407 was prepared and merged. Lastly, we started the integration of console with 29539. The yocto part will come next.

Meta-aws
With the latest virtio-loopback-driver patch that supports both qemu-x86 and qemu-arm64, we are now in line with the execution of virtio-loopback in AWS. We will test the image once Jan-Simon completes the initial support for the AMI creation.

Console & CAN upstreaming
We started preparing a pull request for the vhost-device rust-vmm community which will be ready next week.

Sound:

A new multi-sound device guide is created for Yamaguchi-san to use. It describes all the steps required to reproduce the demo presented during OSS23

GPU:
The work continued on integration testing of the vhost-user-gpu protocol requests. In particular tested the functionality of VHOST_USER_GPU_GET_DISPLAY_INFO, VHOST_USER_GPU_GET_EDID and VHOST_USER_GPU_SCANOUT which are requested by the virtio-gpu driver in order to retrieve and setup the requirements from the display-server side.
The VHOST_USER_GPU_CURSOR_POS/CURSOR_POS_HIDE/CURSOR_UPDATE requests are implemented as dummy functions at this moment. Continued on testing the requests that are carrying the frames-payload from the contrib/vhost-user-gpu application to the virtio-loopback-adapter focused on the VHOST_USER_GPU_UPDATE/DMABUF_UPDATE,VHOST_USER_GPU_SCANOUT and VHOST_USER_GPU_DMABUF_SCANOUT. Successfully tested the acquisition and transfer of DMABUF file-descriptor resources from the vhost-user-gpu application to the adapter.
The following work includes the functional test and benchmark of the vhost-user-gpu support on virtio-loopback-adapter. The benchmark use-case will compare 3D workloads running on virito-loopback and compared with the frame rate achieved by vhost-user-gpu when running with a QEMU VM.

week 21

gerrit reviews:
This week we started with a new version of 29407 (rust implementation of GPIO and RNG), adding CAN and addressing comments. Also we produce important patches for the adapter repository (29398 and 29398) and fixes at kernel side (29492) that aims to build both RefHW and qemu x86 targets. The review of the first one is ongoing, while the other three are now merged. Next is new version of 29407 (v4).

AWS task:
We started the investigations of meta-aws to understand where to put our fingers. The idea is to use EC2 AMI creation feature to build a linux image for AWS that contains virtio loopback binaries, drivers and adapter. Let us know if you have any additional comments or requests for this task.

Virtio-Console:
We established a communication between adapter / vhost-user-console backend. During this communication, the two user-space components exchange virtio features, vhost-user features, queue number, queue size, etc. but they do not exchange any data. During the sub-task, we develop vhost-user-console backend in order to define two simple receive/send queues and use this simple setup to exchange messages with the adapter. Also, the serial console is shown in vhost-device-console terminal and we added send/receive functionality code into vhost-user-console and multiport console feature. After the initial communication between adapter and vhost-user-console device, a new virtio-console (/dev/hvc0) device is being inserted successfully into the kernel. At that moment, by using "getty" for this new device ("hvc0"), messages are sent/received through virtio-console, virtio-loopback-transport, adapter, to vhost-user-console device and vice versa. This results in giving the ability to any user to login and have terminal access through vhost-device-console. The virtio-console multiport feature is added and initializes two additional control queues for the exchange of console console messages. Next week we plan to polish the code for finalization.

GPU:
The work focused on the integration of virtio-gpu.c/virtio-gpu-gl.c on the virtio-loopback-adapter and its dependent functions. Tested the SDL2 /EGL window creation and frame update functionality using pre-recorder frames. Initial tests showed frame updates when the rendering API 'dpy_gfx_update()' is used. Continued on resolving structure dependencies from QEMU that are not ported to the adapter application. This is done in order to not pull all minor components from QEMU sources. Updated the virtio-gpu cursor event requests to dummy implementations since cursor support is not yet reached/tested.
The next steps are to enable and test the handling of contrib/vhost-user-gpu requests related to the Window instantiation (SDL) and the frames update. Also, test the integration of virtio-gpu/gl devices in regard to the parent vhost-user-gpu device on virtio-loopback-adapter. The use-case testing compositor will be Weston.

week 20

gerrit reviews:
Egvirt feature changes (29408) was merged. Review for 29407 (rust implementation of GPIO, I2C, RNG and VSOCK) is ongoing and I hope it will be merged soon. Next we are going to start the activities to push the CAN device. In the meantime we are working to build the sound recipe (currently dealing with rust dependencies), and will come back to you next week.

Virtio-Console:
We are following last week's plan and we started working on the first implementation of the vhost-user-console interface for the virtio-loopback-adapter. This is the part of the adapter that handles the communications between virtio-console and the vhost-user-console backend in rust (not available, it will be implemented from scatch). As a consequence, there is the opportunity and willingness on our side to close the loop with the rust-vmm community and to propose to them the new devices we develop. Let me know if you see any counter-indication on this.
Additionally, a new vhost-user-console device was implemented under adapter which compiles without error and it is ready to be tested with a backend device. An initial draft version of vhost-user-can (under rust-vmm project), is already in place but the device is not mature enough to be compiled and tested with the adapter's component. Next week, we aim to have a first communication between adapter and backend device, exchanging virtio and vhost-user features, queue number, sizes etc.

GPU:
The activity continued on the integration and testing of QEMU's 'hw/display/virtio-gpu.c' and 'hw/display/virtio-gpu-gl.c'devices on virtio-loopback-adapter. Integrated the SDL2/EGL window rendering dependencies of the virtio-gpu device.
Tested the communication of the reworked `qemu_chr_*` API to sockets with success. Resolved dependencies issues on 'virtio-gpu.c/virtio-gpu-gl.c' such as the QTimer and other structures which are not ported from QEMU.Continued on the wiring of the virtio-gpu/gl device to the vhost-user-gpu and the VirtioGPUBase structures on the adapter side.The next steps are the integration of virtio-gpu/virtio-gpu-gl devices support on virtio-loopback-adapter, test the SDL2 window rendering and run the Weston compositor for further testing and benchmarking.

week 19

gerrit reviews:
The first batch of vhost-user devices was pushed (29407). This includes rust implementation of GPIO, I2C, RNG and VSOCK, taken from meta-virtualization.
Also, a small restructuring of the egvirt feature has been proposed (29408). Reviews are ongoing.
Next week we plan to work on recipes for vhost-user-sound and vhost-user-can (more details below on the latter).

CAN:
This week we finalized the implementation of vhost-user-can on top of virtio-loopback. The code was polished and aligned with QEMU and Rust-vmm code style. Next week we are going to prepare yocto recipes. Source files will be then included in meta-egvirt. When .rs files will be merged in meta-egvirt, we will communicate with the rust-vmm community to assess their interest in merging this device.

Virtio-Console:

We started working on the vhost-user-console support and after some investigations we prepared an initial plan. In general, (similarly to what happened for CAN) there is no vhost-user device available publicly, and the plan includes the development of a rust vhost-user-can device that will be shared with the rust-vmm community. Plan milestones:
1) Implement vhost-user-console Adapter interface (deadline 24/11/2023) - Based on other adapter's device implementation and QEMU's virtio-console device.
2) Vhost-user-can backend in Rust (deadline 08/12/2023) - Implement vhost-user-can based on vhost-device-can and QEMU's virtio-console device.
- [vhost-device](https://github.com/rust-vmm/vhost-device/tree/main)
3) Testing & Prepare the release: (deadline 15/12/2023) - Polish/Push the epsilon-release version by the addition of vhost-user-console

GPU:
The work carried on the integration of vhost-user-gpu/virtio-gpu devices on virtio-loopback-adapter. Completed the porting of `qemu_chr_*` API functions to unix-socket API for `vhost-user-gpu.c`. Integrated the virtio-gpu-base-base device and reworked the QEMU API-accessor macros /functions.
Performed functional tests with virtio-loopback-adapter and `contrib/vhost-user-gpu` application upto the point where the virtual-gpu is instantiated. Continued on further integration of QEMU's `hw/display/virtio-gpu.c`and `hw/display/virito-gpu-gl.c` as dependencies for an OpenGL-accelerated virtio-gpu device support. Integrated the SDL rendering backend to support the visual representation of the frames on a Wayland compositor. Next steps include the integration the virtio-gpu devices support on virtio-loopback-adapter and perform functional test with a Wayland compositor (Weston).

week 18

GPU:

The task continues on finalizing the display rendering and vhost/virtio device model support on virtio-loopback-adapter.Extracted partially the Type and Object device systems of QEMU as required for the instantiation of the vhost-user-gpu and virtio-gpu devices. Reworked the QEMU Character-Device API `qemu_chr_*` found in `vhost_user_gpu.c` and `virtio-gpu-base.c`and replaced it with conventional unix-socket API. Continued on the wiring of the top-level rendering framework and the instantiation of the window that hosts the frames as instructed by the VhostUserGPU protocol.

The next steps include the integration of the display rendering components with the vhost-user- and virtio-gpu device models.
Additionally, test 3D workloads on a Wayland compositor such as Weston.

CAN:
CAN/CANFD messages are being exchanged between the virtio-can and vhost-user-can backend (+ "vcan0").
This week, we developed the send/receive functionality code into vhost-user-can for sending/receiving messages to real hardware CAN device and add support for CANFD messages. After the initial communication between adapter and vhost-user-can device, a new virtio-can device is being inserted successfully into the kernel. At that moment, by using "can-utils", CAN/CANFD messages are sent through virtio-can, virtio-loopback-transport, adapter, vhost-user-can to "vcan0" device and vice versa. Next week we plan to polish the code for finalization.

week 17

GPU:

Activity continues aiming at finalizing the port of the hw/virtio/vhost-user-gpu.c dependencies for the rendering framework and the vhost-user-gpu protocol. On virtio-loopback-adapter we added support for the VirtIOGpu device that implements the display framework backend of QEMU. Isolated and extracted the top-level GTK hooks and the display-listener used by QEMU. Next steps include finalizing the vhost-user-gpu support on the adapter and the port of display dependencies from QEMU.
As mentioned last week, there is a small delay on this task, however we plan to produce a first delivery before the next SDV call.

CAN:
This week we established communication between adapter / vhost-user-can backend and added most of CAN characteristics in the backend. During this communication, the two user-space components exchange virtio features, vhost-user features, queue number, queue size, etc. but they do not exchange any data. During the sub-task, we develop vhost-user-can backend in order to receive/send messages to adapter and integrate CAN related missing parts into the backend implementation, which were taken from QEMU's virtio-can device. Next week our effort will be focused on operating this whole structure with virtio-can driver and mainly on the data exchange process between the two components

week 16

GPU:

This week we identified the GPU rendering issues we were experiencing as dependent on the virtio-gpu driver to allow access on the framebuffer resource. There is then a need to remove kernel-space dependencies and fully render GPU graphics in user-space inside the adapter application. As a result, we are now working in that direction, updating the adapter's vhost-user-gpu support with the SDL rendering pipeline of Qemu. This could lead to a small delay in the delivery of the GPU support (initially planned for Oct. 31st). We will provide more info with a new deadline next week.

CAN:
Finish an initial draft version of vhost-user-can under the adapter. A new vhost-user-can device was implemented under adapter which compiles without error and it is ready to be tested with a backend device. In parallel, continue working with vhost-user-can under rust-vmm project. An initial draft version, was already in place but the device is not mature enough to be compiled and tested with adapter's component. Next week, we target to have an first communication between adapter and backend device, exchanging virtio and vhost-user features, queue number, sizes etc.

week 15

GPU:

Updated the virtio-loopback-adapter GPU's support and split the frames-rendering part to a standalone application. This is intended to result in a viewer-application that uses standardized DRM/KMS APIs to retrieve and render frames from the character device that is probed by the virtio-gpu driver. Performed tests to verify the virtio-gpu probed driver state and the message exchanges with vhost-user-gpu. Tested the virtqueues management and the events/irqs received by the virtio-loopback-adapter threads with success.
Next steps include completing and resolving the issues on the viewer-application for virtio-gpu.

CAN:
We are following last week's plan and we started working on the first implementation of the vhost-user-can interface for the virtio-loopback-adapter. This is the part of the adapter that handles the communications between [virtio-can](https://lwn.net/Articles/934187/) with the vhost-user-can backend in rust (will be implemented in parallel). As of today, since there is no vhost-user-can front-end device in QEMU project, the development will be based on older adapter's device (ex. GPIO) and the [virtio-can](https://lwn.net/Articles/934187/) QEMU device which is still under RFC phase. Next week we will continue working on the adapter's component, while starting to build an initial vhost-user-can backend device under rust-vmm.

week 14

GPU:
We are working on the partial rendering issue we have with the glmark2 application. In the debug we are particularly focusing on the virtio-loopback-adapter and the virtqueues notification event mechanism which bridges the vhost-user-gpu and the virtio-gpu driver.

Next steps include resolving the rendering issue and polishing the changes on virtio-loopback-adapter and virtio-loopback-driver.

CAN:
We started working on the CAN support and after some investigations we prepared an initial plan. In general, there is no vhost-user device available publicly, and the plan includes the development of a rust vhost-user-can device that will be shared with the rust-vmm community.
Plan milestones:
1) Implement Vhost-user-sound Adapter interface (deadline 23/10/2023)

- Based on other adapter's device implementation: [virtio-can RFC](https://lwn.net/Articles/934187/)
2) Vhost-user-can backend in Rust (deadline 01/12/2023)
- Implement vhost-user-can based on vhost-device-rng & vhost-device-gpio
- [vhost-device](https://github.com/rust-vmm/vhost-device/tree/main)

3) Testing & Prepare the release: (deadline 08/12/2023)
- Polish/Push the epsilon-release version by the addition of vhost-user-can

week 13

GPU:
We moved the development environment on a host (we were working inside a guest) and we resolved the latest stability issues. Now we are able to load/unload the driver multiple times etc. Rendering is still partial with the glmark2 application. It needs more investigation that will be done next week.

Sound:

Vhost-user-sound device is added in virtio-loopback, tested and pushed (epsilon-release).
- [adapter](https://git.virtualopensystems.com/virtio-loopback/adapter_app/-/tree/epsilon-release)
- [loopback driver](https://git.virtualopensystems.com/virtio-loopback/loopback_driver/-/tree/epsilon-release)
- [vhost-device](https://git.virtualopensystems.com/virtio-loopback/vhost-user-rng-rust/vhost-device/-/tree/epsilon-release) (sound & gpio)

Note: The vhost-user-sound & vhost-user-gpio backend implementations were taken by the developer repo [vhost-device](https://github.com/virtio-sound/vhost-device) (since vhost-user-sound is under development).

We integrated new sound and GPIO devices in the epsilon release and we will be pushing this new version of the 29225 patches early next week.

Week 12

GPU:
The effort on the GPU task was spent on finalizing the visualization of 3D applications. Continued the tests with the glmark2 OpenGL application. At the current stage the visualization is not yet stable and we are working to identify the issue.
Next step is to find the current issue and finalize window rendering..

Sound:

In the context of the audio tests, we updated the adapter and loopback driver exchange notifications and interrupts. After the updates, the behavior of the sound device seems to be stable, and we able to get audio out of the speakers when using the "pipewire" backend. Tests were applied for the rest of the vhost-user-sound's backends as "null" and "alsa".
Next week we will continue testing the virtio-sound behavior on top of virtio-sound, and start preparing a preliminary release of the code under the "epsilon-release" branches.

GPIO:

Vhost-user-gpio device is added in virtio-loopback, tested and pushed (epsilon-release).
  - [adapter](https://git.virtualopensystems.com/virtio-loopback/adapter_app/-/tree/epsilon-release)
  - [loopback driver](https://git.virtualopensystems.com/virtio-loopback/loopback_driver/-/tree/epsilon-release)
The vhost-user-gpio backend implementation was taken by the upstream [rust-vmm repository](https://github.com/rust-vmm/vhost-device/tree/main).
Next step is to embed the device in the yocto recipe.

Week 11

GPU:
Continued the work on the visualization of virtio-gpu framebuffers. Tested the merge of static contextes (background framebuffer) and GL contextes from the 3D pipeline with a test application. Started to test the example 3D applications from mesa-demos to check for incompatibilities. Next work includes the finalization of visualization/window rendering and getting information about the performance with a GPU benchmark application.

Sound:
As mentioned last week, we are working in parallel to evaluate two scenarios:
1) Testing vhost-user-sound with Qemu:
Started testing vhost-user-sound with both "ALSA" and "Pipewire" backend and Qemu. We weren't able to make it work, we tried both linaro's and redhat's versions of QEMU. On "pipewire" case, when we tried to play audio with "aplay audio.wav" (or mpg123 audio.mp3), we get a short sound from the speaker and an debug messages "Dropping IOmessage". Lastly, on the "ALSA" case, an error message is being printed, followed by the same "Dropping IOmessage" log and the stream is closed.
2) Testing vhost-user-sound on virtio-loopback:
Also in this scenario, we tried to output audio by running vhost-user-sound on top of virtio-loopback. By using the "pipewire" backend and running "aplay audio.wav" we were able to get a full song played on the speakers. During the execution, logs "Dropping IOmessage" were printed, audio came out only from one of the speakers and we noticed that the stream was not closed successfully after the song's end. This might be caused due to the "stop" message doesn't reach vhost-user-sound successfully. "ALSA" backend did not yet work successfully also on "virtio-loopback" case.
Next week we will continue to investigate the ALSA and Pipewire backends on both "QEMU - vhost-user", "Virtio-loopback - vhost-user" scenarios. Our first priority, would be to understand which of the errors noticed are derived by virtio-loopback implementation and correct them, targeting to play a sound stream on the speakers successfully.

GPIO:
We started polishing the new adapter's gpio device and working on a new release.
Next week we plan to finalize and push the gpio release on our git servers. The gerrit patches will arrive the week after.

Week 10

Activity

GPU:
Worked on the visualization part of the virtio-gpu DRM device. Continued the investigation on how to combine framebuffers from multiple sources.
Explored the QEMU display handling code (high-level) for potential similarities and implementation specific constraints.

Sound:

Testing vhost-user-sound on virtio-loopback: This week we were able to solve the issue and eliminate any behavioral differences between the scenarios "QEMU - vhost-user-sound" and "Virtio-loopback - vhost-user-sound". Seems that on virtio-loopback case, for big audio files we observe more stable behavior when SMP is enabled. We continue our investigation by start testing sound device with ALSA backend where we found some errors printed from the device side when try to play an audio file.
Next week we will continue to investigate the ALSA case and in parallel start testing sound with "Pipewire" backend which is still under active development to evaluate the current status of the device.

GPIO:
Testing vhost-user-gpio on virtio-loopback: Continue developing the vhost-user-gpio frontend into the adapter and testing it with "gpio-mockup" pseudo device in the kernel. We are able to observe notification and data being exchanged between virtio-gpio and vhost-user-gpio. In more details, we are able to read GPIO lines and bits by using "gpioget" command, but not able to change those values ("gpioset"). After a short investigation, "gpioset" was not able to change gpio value even directly on "gpio-mockup" device (no virtio-loopback in place), so this proves was a issue unrelated to the virito-loopback solution.
Next week we plan to continue testing the virtio-gpio device, and start polishing the implementation if no new issues occur.

Week 9

GPU:
Worked on the finalization effort for enabling the virtio-gpu DRM renderer device. Created a simple OpenGL workload application to test the functionality of the DRM renderer device and the handling of GL calls by 'virgl'.
Successfully managed to run the 3D application without exceptions anywhere from the software-stack. Verified the GL operation by tracing the calls on vhost-user-gpu and the handling code in 'virgl'. Begin to extend the software stack in order to visualize the frames of the framebuffer of the virtio-gpu device. At the moment this part will be integrated on the 'Adapter' application.
The next steps are to continue the development of the visualization code for the virtio-gpu frames, and explore how to deal with multiple contexts when the 3D acceleration is enabled.

sound:
This week we tested vhost-user-sound with the new adapter sound interface, comparing the behavior of vhost-user-sound between the scenarios a) virtio-loopback (non-hypervisor) and b) QEMU (host-guest) to test similarities and differences. During the tests we noticed that some notifications from multiple virtqueues coming from the kernel collided. We updated the notification mechanism for multiple virtqueues between the driver and adapter to make sure that all of them are delivered and with the same order.
Next week we will continue to investigate the above behavior and go one step closer to finalize the vhost-user-sound adapter frontend before moving to experiment with other sound backend as "alsa" and "pipewire".

gpio:

We continued investigations on the linux kernel side (`gpio_virtio.ko`) and on the vhost-user-gpio device. Additionally, we added support for a new generic device in the adapter (to be extended as a gpio device next week). The development environment has been prepared, and it includes the gpio-mockup kernel module, which is going to be used to test the implementation.

Next week, we plan to extend the existing adapter support to gpio (today it is just a generic device).

Week 8

Activity

GPU:

Worked on the Adapter application to debug and trace the enablement of the virtio-gpu DRM Renderer device. The device is part of the virtio-gpu driver, handles memory requests from the user-space driver library, and exposes the render node via '/dev/dri/renderDxxx' Updated the Adapter to inform the 'virtio-mmio' driver with the correct parameters in order to register the virtio-gpu DRM driver with the renderer feature enabled. Successfully made the virtio-gpu driver to register both the display and renderer devices.

Our plan for next week are on stabilizing the virtio-gpu DRM driver with the renderer feature and test an OpenGL application

sound:

Development of the first draft of the virtio-loopback-adapter sound interface complete. Now, a new virtio-sound device is being registered successfully into the kernel and the virtio-loopback-adapter sound interface is able to communicate (exchange notifications) with the vhost-user-sound device. The vhost-user-sound backend when using the "null" backend seems to be able to access the shared data space without any error and read data. This proves that the compatibility layer for rust devices developed the previous week works as expected also for vhost-user-sound.

Next steps will be to check the validity of the data read by vhost-user-sound from the share data space, and soon after, start experimenting with alsa and pipewire.

gpio:

We started the activity with investigations on the current gpio driver status (vhost user device available, developed in rust).

Next week we plan to start adding gpio support in the adapter application.

Week 7

Activity

GPU:

This week we focused on running the weston compositor on virtio-gpu with virtio-looback. Tested the user-space DRM char-devices with libdrm examples and proceeded on running the weston compositor. At this moment, there is no visual output for the frames rendered by virtio-gpu, thus the frames are checked by dumping the contents of the framebuffer.

Next week: the steps are to enable the 3D acceleration on vhost-user-gpu and make sure that the virtio-gpu DRM device probes the renderer device as well. More testing will be done by creating an OpenGL workload application to debug & trace the overall stack.

sound:

Continue working on the adapter sound front-end and upgrade the data sharing mechanism in order to exchange data with vhost-user-sound backend (in Rust).

The target of this week, was to: a) create a new compatibility layer in the adapter so vhost-user sound device can access the shared space without the requirement of any modification of vhost-user-sound, b) continue the development of adapter vhost-user-sound front-end, so a new linux virtio-sound device will be register successfully at the end of this process. The updated adapter version (a) has the new compatibility layer integrated, and it was tested with vhost-user-rng device in rust, and in C to cross-validate it (testing it with vhost-user-sound will be done later at a later stage). About the "device registration" objective (b), virtio-sound in linux seem to not register a new device yet, so the adapter sound front-end part still needs further development.

Next week: we will continue working on the front-end sound part and target an initial version of the adapter which will be ready to register the virtio sound device in the kernel successfully.

Week 6

Activity

sound: Established an initial communication between adapter and vhost-user-sound backend. During this communication, the two user-space components exchange virtio features, vhost-user features, queue number, queue size, etc. but they do not exchange any data. We developed some missing parts of the adapter in order to handle all messages (excluding for now data related messages) i.e., send/receive to/from the vhost-user-sound device using the "null" backend.

Plan for next week Next week our effort will be focused on the data exchange process between the two components (adapter, vhost-user-sound backend).

Week 5

Activity

GPU:
We continued the effort on enabling all features of the virtio-gpu DRM driver. At this stage 'vhost-user-gpu' runs without the 'virgl' flag, thus meaning that only the display part of the virtio-gpu DRM driver is enabled.
Started testing the functionality of the virtio-gpu driver over virtio-loopback. Primarily, the tests are using the libdrm library to ensure the expected functionality from the virtio-gpu DRM driver.
The foreseen development and testing plan consist of running the weston compositor on the virtio-gpu and enabling the 3D translation path on vhost-user-gpu, Adapter and the virtio-gpu driver.

sound:
We are working at the first implementation of the vhost-user-sound interface for the virtio-loopback-adapter. This is the part of the adapter that handles the communications with the vhost-user-sound backend. Now the adapter successfully creates an instance of the sound device, and starts the message exchange with the vhost-user backend device. As of today, we based this development on the RedHat solution because we found it as more complete than Linaro's implementation (see weekly report 4 for more info). We are constantly monitoring the vhost-user-sound developments to make sure we do not miss any update and minimize the impact of changes on our work with virtio-loopback.

Plan for next week Next week we will continue working on the adapter component, while we are testing its communication with vhost-user-sound backend and developing any missing parts.

Week 4

Activity

GPU: We continued the effort on enabling all features of the virtio-gpu DRM driver. At this stage 'vhost-user-gpu' runs without the 'virgl' flag, thus meaning that only the display part of the virtio-gpu DRM driver is enabled. Started testing the functionality of the virtio-gpu driver over virtio-loopback. Primarily, the tests are using the libdrm library to ensure the expected functionality from the virtio-gpu DRM driver.
The foreseen development and testing plan for next week consists of running the weston compositor on the virtio-gpu and enabling the 3D translation path on vhost-user-gpu, Adapter and the virtio-gpu driver.

Sound: This week we explored the initialization procedures for devices with existing support on 'virtio-loopback' (input, blk, rng), and started a preliminary skeleton implementation for 'vhost-user-sound' on the 'Adapter' application.

Moreover, we continued the investigations and we found two alternative implementations of vhost-user-sound (one from RedHat, and one from linaro). Since none of the two is upstream, we will try to understand what's the most advanced and the most likely to be accepted upstream. Similarly to what we prepared with GPU, the development plan for sound is the following:

Part 1: Work on the virtio-loopback Adapter application
Add support for the vhost-user-sound device on the adapter application.
This will handle the new command-line configuration for vhost-user-sound, handle the socket connection with 'vhost-user-sound' backend and to instantiate/load the 'virtio-sound' driver module At this stage it will be possible to start and communicate with the rust backend device, but no way to hear the output on the speakers.
Estimation: ~4.5 weeks

Part 2: Update vhost-user-sound device from rust-vmm
This task targets to fill any development gaps existing at the time of vhost-user-sound backend and establish a successful communication with the Adapter and eventually with virtio-sound. The above-mentioned repositories already implement a vhost-user-sound backend in rust, although they might not be very stable as they are under development. This task should carefully evaluate the two to decide which way to go and integrate one of the two as AGL solution. Fixes on top of the selected existing solution, if needed, will be implemented.
Estimation: ~4.5 weeks

Part 3: Integrate/testing vhost-user-sound, virtio-loopback with Pipewire/ALSA
This task will focus on establishing a successful communication between virtio-sound and Pipewire via virtio-loopback. Putting together parts 1 and 2, the outcome of this task is an application will be able through virtio-loopback and vhost-user device to communicate with Pipewrire/Alsa and output to the speakers.
Estimation: ~3 weeks

Part 4: Integration/testing with virtio-loopback upstream and AGL yocto layers
This task targets the upstreaming effort to update the upstream version of virtio-loopback.
Estimation: 1 week

Week 3

Activity

GPU: We focused on the Adapter app debugging the interactions and data shared on the socket interface connected to the vhost-user-gpu. We managed to resolve the registration of the Virtio-GPU DRM driver issue mentioned last week. Now the virtio-GPU DRM driver is loaded correctly by the loopback-driver.
We are now about to test on the Virtio-GPU DRM character device to verify and match the functionality that is needed. At the same time, we started the activities related to the sound as discussed during the last EG call.

Plan for next week Plan for next week is to continue the work for testing the DRM driver, as well as to progress with the sound.

Week 2

Activity

GPU: The activities advancement for this week are mostly on the 'Adapter' application. We progressed with the development of the virtio-gpu interface on the 'Adapter' and investigated the code-path until the virtio-loopback driver requests to register the virtio-gpu driver. At this stage the vi rtio-gpu driver registration takes place with invalid parameters and the DRM driver is not loaded correctly. We then continued on debugging the communication data from 'vhost-user-gpu' to the 'Adapter' as well as the features set passed when running the 'vhost-user-gpu' with the '-virgl' flag.

Plan for next week The next steps include the successful registration of the virtio-gpu driver, continue the development of the 'Adapter' and initiate a plan to test the functionality of the virtio-gpu driver. The goal is to run a compositor (weston) on the character-device exposed by the virtio -gpu and stress the functionality of the solution. Similar tests will be carried on at a lower-API level using the 'libdrm' library.

Week 1

Activity

GPU: *we* continued the investigation on 'vhost-user-gpu' and the vhost-user protocol. We started the development of the virtio-gpu agnostic code on the 'Adapter' application. Explored the communication messages sent by the 'vhost-user-gpu' application to the 'Adapter' via the socket interface as well as messages from devices with existing support (virtio-rng) to compare. Traced the feature-set provided by 'vhost-user-gpu', the vhost-user handling code and continued with the support of the gpu interface on the 'Adapter' application.

Plan for next week is to investigate the virtio-gpu driver registration and continue the development of virtio-gpu specific code on the 'Adapter' application.

Week 0 (started 2023-07-03)

ActivityT

GPU:his week we started the investigations on the 'vhost-user-gpu' implementation of Qemu and its socket interface. This qemu component provides the device emulation for 'virtio-gpu' and allows an external application to connect via socket (virtio-loopback 'Adapter' in our case). Moreover, we explored the initialization procedures for devices with existing support on 'virtio-loopback' (input, rng), and started a preliminary skeleton implementation for 'vhost-user-gpu' on the 'Adapter' application. As a result of these investigations, we prepared a development plan for the next weeks:

virtio-GPU development plan

- Part 1: decouple the Virtio-GPU device from the QEMU

  This task targets to explore and understand the virtio-gpu device (vhost-user-gpu) from the QEMU project. The functionality is packed in an application and later used via socket by the adapter application. This includes the graphics (rendering) part of the GPU as well as the display portion.

  Estimation: ~4 weeks

- Part 2: Work on the virtio-loopback Adapter application

  Add support for the virtio-gpu device on the adapter application.
  This will handle the new command-line configuration for virtio-gpu, handle the socket connection with 'vhost-user-gpu' and to instantiatiate/load the 'virtio-gpu' driver module At this stage it will be possible to start and communicate with the GPU device, but no way to see the output on a screen/window.

  Estimation: ~4 weeks

- Part 3: Wiring the adapter with virglrenderer and integration with the display server

  By concept, the GPU commands, data are proxied and handled by the virglrenderer library. On this task we are wiring the virglrenderer context with the adapter application to allow the client-virtio-GPU context to be forwarded to the real-GPU context. As a proof-of-concept the display server will be a Wayland based compositor.

  Estimation: ~4 weeks

- Part 4: Integration/testing with virtio-loopback upstream and AGL yocto layers

  Estimation: ~1 weeks

Plan for next week is to go deeper in Part 1 with the addition of the 'vhost-user-gpu' interface to the 'Adapter' and the instantiation of the 'virtio-gpu' driver by the loopback kernel module.