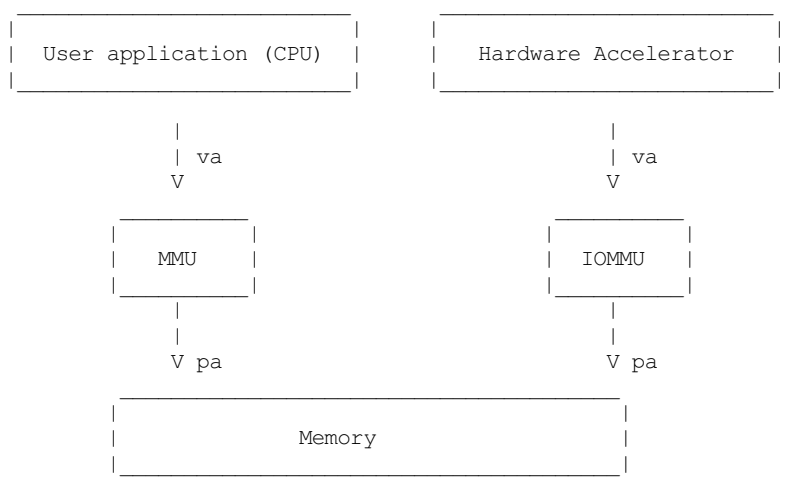
Introduction of Uacce

Uacce (Unified/User-space-access-intended Accelerator Framework) targets to provide Shared Virtual Addressing (SVA) between accelerators and processes. So accelerator can access any data structure of the main cpu. This differs from the data sharing between cpu and io device, which share only data content rather than address. Because of the unified address, hardware and user space of process can share the same virtual address in the communication. Uacce takes the hardware accelerator as a heterogeneous processor, while IOMMU share the same CPU page tables and as a result the same translation from va to pa.

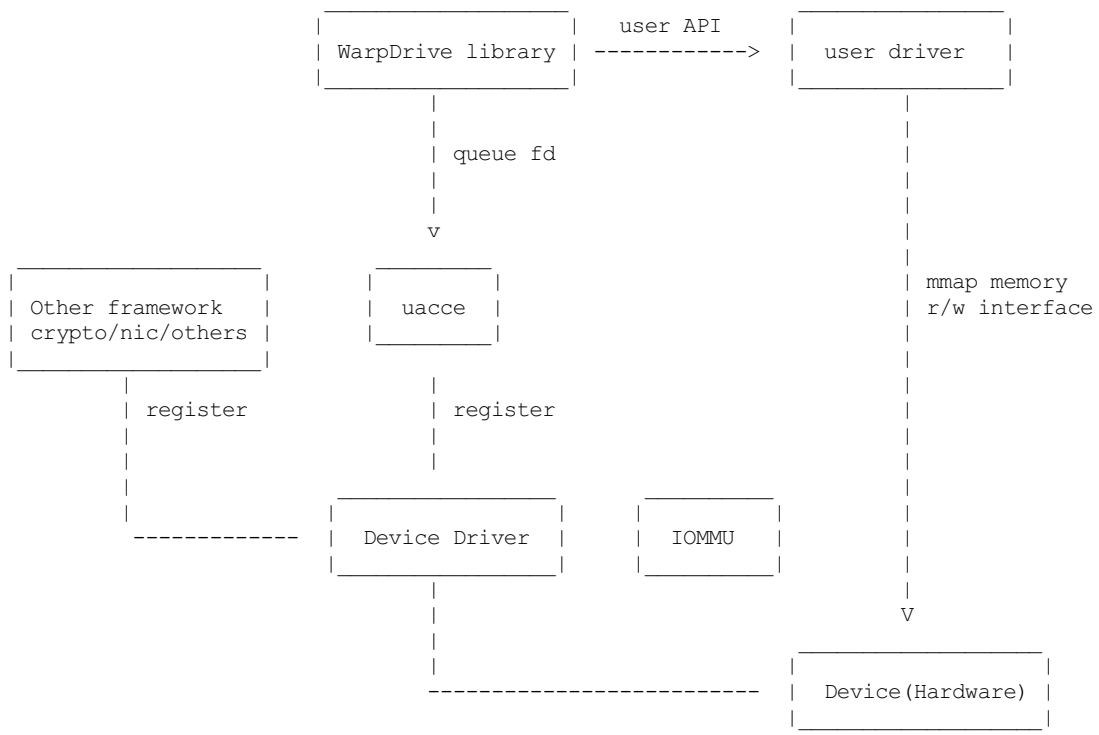


Architecture

Uacce is the kernel module, taking charge of iommu and address sharing. The user drivers and libraries are called WarpDrive.

The uacce device, built around the IOMMU SVA API, can access multiple address spaces, including the one without PASID.

A virtual concept, queue, is used for the communication. It provides a FIFO-like interface. And it maintains a unified address space between the application and all involved hardware.



How does it work

Uacce uses mmap and IOMMU to play the trick.

Uacce creates a chrdev for every device registered to it. New queue is created when user application open the chrdev. The file descriptor is used as the user handle of the queue. The accelerator device present itself as an Uacce object, which exports as a chrdev to the user space. The user application communicates with the hardware by ioctl (as control path) or share memory (as data path).

The control path to the hardware is via file operation, while data path is via mmap space of the queue fd.

The queue file address space:

```
 /**
 * enum uacce_qfrt: qfrt type
 * @UACCE_QFRT_MMIO: device mmio region
 * @UACCE_QFRT_DUS: device user share region
 */
enum uacce_qfrt {
        UACCE_QFRT_MMIO = 0,
        UACCE_QFRT_DUS = 1,
};
```

All regions are optional and differ from device type to type. Each region can be mmapped only once, otherwise -EEXIST returns.

The device mmio region is mapped to the hardware mmio space. It is generally used for doorbell or other notification to the hardware. It is not fast enough as data channel.

The device user share region is used for share data buffer between user process and device.

The Uacce register API

The register API is defined in uacce.h.

```
struct uacce_interface {
  char name[UACCE_MAX_NAME_SIZE];
  unsigned int flags;
  const struct uacce_ops *ops;
};
```

According to the IOMMU capability, uacce_interface flags can be:

```
/**
 * UACCE Device flags:
 * UACCE_DEV_SVA: Shared Virtual Addresses
 *              Support PASID
 *              Support device page faults (PCI PRI or SMMU Stall)
 */
#define UACCE_DEV_SVA               BIT(0)

struct uacce_device *uacce_alloc(struct device *parent,
                                 struct uacce_interface *interface);
int uacce_register(struct uacce_device *uacce);
void uacce_remove(struct uacce_device *uacce);
```

uacce_register results can be:

a. If uacce module is not compiled, ERR_PTR(-ENODEV)
b. Succeed with the desired flags
c. Succeed with the negotiated flags, for example

uacce_interface.flags = UACCE_DEV_SVA but uacce->flags = ~UACCE_DEV_SVA

So user driver need check return value as well as the negotiated uacce->flags.

The user driver

The queue file mmap space will need a user driver to wrap the communication protocol. Uacce provides some attributes in sysfs for the user driver to match the right accelerator accordingly. More details in Documentation/ABI/testing/sysfs-driver-uacce.