IO PAGE FAULT FOR ALL

Linux Plumbers Conference

Joel Granados j.granados@samsung.com

Samsung GOST

September 2024

AGENDA

- Motivation
- Plumbing
- Setup Code
- Using Code
- Forward

MOTIVATION?

- Increasing Possibilities for
  - user-space driver driven device verification
  - Avoid memory pinning
- User space testing frameworks

PLUMBING

- Intel IOPF is already in the Linux Kernel
  - Targeted/Under `INTEL_IOMMU_SVM`
  - PASID capability is checked
- We move it out of `INTEL_IOMMU_SVM`
  - Into `INTEL_IOMMU`
  - Drop PASID check
  - IOPF regardless of SVM or PASID

PLUMBING

- Moved to drivers/iommu/intel/prq.c
- Remove pasid present check
- Move IOPF Kconfig to INTEL_IOMMU
- Enable PRI on iommufd_hwpt_alloc
- Remove extended capability check for PASID

iommu: Enable user space IOPFs in non-PASID and non-svm cases

PLUMBING

Moved to drivers/iommu/intel/prq.c

- struct page_req_dsc
- intel_drain_pasid_prq
- intel_page_response
- intel_finish_prq
- intel_enable_prq
- prq_event_thread
- intel_prq_report
- prq_to_iommu_prot
- handle_bad_prq_event
- is_cannonical_address

iommu: Enable user space IOPFs in non-PASID and non-svm cases

PLUMBING

Remove pasid present check

```
diff --git a/drivers/iommu/intel/prq.c
@@ @@ static irqreturn_t prq_event_thread(int irq, void
        *d)
    while (head != tail) {
        req = &iommu->prq[head / sizeof(*req)];
        address = (u64)req->addr << VTD_PAGE_SHIFT;

-         if (unlikely(!req->pasid_present)) {
-             pr_err("IOMMU: %s: Page request without
        PASID\n",
-                    iommu->name);
-bad_req:
-             handle_bad_prq_event(iommu, req,
        QI_RESP_INVALID);
```

iommu: Enable user space IOPFs in non-PASID and non-svm cases

PLUMBING

Move IOPF Kconfig to INTEL_IOMMU

```
diff --git a/drivers/iommu/intel/Kconfig
@@ @@ config INTEL_IOMMU
        select IOMMU_IOVA
+       select IOMMU_IOPF
        select IOMMUFD_DRIVER if IOMMUFD
@@ @@ config INTEL_IOMMU_SVM
        select IOMMU_SVA
-       select IOMMU_IOPF
        help
```

iommu: Enable user space IOPFs in non-PASID and non-svm cases

PLUMBING

Enable PRI on iommufd_hwpt_alloc

```
diff --git a/drivers/iommu/intel/iommu.c
@@ @@ intel_iommu_domain_alloc_user(struct device *dev, u32
        flags,
        if (flags &
-           (~(IOMMU_HWPT_ALLOC_NEST_PARENT |
        IOMMU_HWPT_ALLOC_DIRTY_TRACKING)))
+           (~(IOMMU_HWPT_ALLOC_NEST_PARENT |
        IOMMU_HWPT_ALLOC_DIRTY_TRACKING
+              | IOMMU_HWPT_FAULT_ID_VALID)))
                return ERR_PTR(-EOPNOTSUPP);
diff --git a/drivers/iommu/iommufd/hw_pagetable.c
@@ @@ iommufd_hwpt_paging_alloc(struct iommufd_ctx *ictx,
        struct iommufd_ioas *ioas,
        const u32 valid_flags = IOMMU_HWPT_ALLOC_NEST_PARENT
```

iommu: Enable user space IOPFs in non-PASID and non-svm cases

PLUMBING

Remove extended capability check for PASID

```
diff --git a/drivers/iommu/intel/iommu.c
@@ @@ static void free_dmar_iommu(struct intel_iommu *iommu)

-       if (pasid_supported(iommu)) {
-               if (ecap_prs(iommu->ecap))
-                       intel_finish_prq(iommu);
-       }
+       if (ecap_prs(iommu->ecap))
+               intel_finish_prq(iommu);
 }
@@ @@ static int __init init_dmars(void)

-               if (pasid_supported(iommu) && ecap_prs(iommu-
        >ecap)) {
```

iommu: Enable user space IOPFs in non-PASID and non-svm cases

SETUP CODE!

```
// bdf : Device ID (BUS,DEVICE,FUNCTION)
// iommufd : iommufd file descriptor
// ioas : target ioas
void setup(char *bdf, int iommufd, struct iommu_ioas
        *ioas) {
    // 1. Bind Dev to IOMMUFD
    // 2. Attach Device to an IOAS
    // 3. Create Fault Queue Allocation
    // 4. Allocate FAULT VALID HWPT
    // 5. Re-attach to FAULT VALID HWPT
}
```

https://github.com/SamsungDS/libvfn/blob/iommufd-fault-queue/examples/iopf.c

SETUP CODE!

```c
// bdf : Device ID (BUS,DEVICE,FUNCTION)
// iommufd : iommufd file descriptor
// ioas : target ioas
void setup(char *bdf, int iommufd, struct iommu_ioas
        *ioas) {
    // 1. Bind Dev to IOMMUFD
    char *path = NULL;
    char VFIO_ID = pci_get_vfio_id(bdf);
    int devfd = open('/dev/vfio/devices/VFIO_ID',
        O_RDWR);

    struct vfio_device_bind_iommufd bind = {
        .argsz = sizeof(bind),
        .flags = 0,
```

https://github.com/SamsungDS/libvfn/blob/iommufd-fault-queue/examples/iopf.c

SETUP CODE!

```c
// bdf : Device ID (BUS,DEVICE,FUNCTION)
// iommufd : iommufd file descriptor
// ioas : target ioas
void setup(char *bdf, int iommufd, struct iommu_ioas
        *ioas) {
    // 1. Bind Dev to IOMMUFD
    // 2. Attach Device to an IOAS
    struct vfio_device_attach_iommufd_pt attach_data =
        {
        .argsz = sizeof(attach_data),
        .flags = 0,
        .pt_id = ioas->id,
    };
    ioctl(devfd, VFIO_DEVICE_ATTACH_IOMMUFD_PT,
```

https://github.com/SamsungDS/libvfn/blob/iommufd-fault-queue/examples/iopf.c

SETUP CODE!

```c
// bdf : Device ID (BUS,DEVICE,FUNCTION)
// iommufd : iommufd file descriptor
// ioas : target ioas
void setup(char *bdf, int iommufd, struct iommu_ioas
        *ioas) {
    // 1. Bind Dev to IOMMUFD
    // 2. Attach Device to an IOAS
    // 3. Create Fault Queue Allocation
    struct iommu_fault_alloc fault = {
        .size = sizeof(fault),
        .flags = 0,
    };
    ioctl(iommufd, IOMMU_FAULT_QUEUE_ALLOC, &fault);
    // 4. Allocate FAULT VALID HWPT
```

https://github.com/SamsungDS/libvfn/blob/iommufd-fault-queue/examples/iopf.c

SETUP CODE!

```c
// bdf : Device ID (BUS,DEVICE,FUNCTION)
// iommufd : iommufd file descriptor
// ioas : target ioas
void setup(char *bdf, int iommufd, struct iommu_ioas
        *ioas) {
    // 1. Bind Dev to IOMMUFD
    // 2. Attach Device to an IOAS
    // 3. Create Fault Queue Allocation
    // 4. Allocate FAULT VALID HWPT
    struct iommu_hwpt_alloc fault_cmd = {
        .size = sizeof(fault_cmd),
        .flags = IOMMU_HWPT_FAULT_ID_VALID,
        .dev_id = bind.out_devid,
        .pt_id = ioas->id,
```

https://github.com/SamsungDS/libvfn/blob/iommufd-fault-queue/examples/iopf.c

SETUP CODE!

```c
// bdf : Device ID (BUS,DEVICE,FUNCTION)
// iommufd : iommufd file descriptor
// ioas : target ioas
void setup(char *bdf, int iommufd, struct iommu_ioas
        *ioas) {
    // 1. Bind Dev to IOMMUFD
    // 2. Attach Device to an IOAS
    // 3. Create Fault Queue Allocation
    // 4. Allocate FAULT VALID HWPT
    // 5. Re-attach to FAULT VALID HWPT
    struct vfio_device_attach_iommufd_pt attach = {
        .argsz = sizeof(attach),
        .flags = 0,
        .pt_id = fault_cmd.out_hwpt_id;
```

https://github.com/SamsungDS/libvfn/blob/iommufd-fault-queue/examples/iopf.c

USING CODE!

```
{
    // 1. Poll IOPF File Descriptor
    struct iommu_hwpt_pgfault pgfault;
    while (read(fault.out_fault_fd,
                &pgfault, sizeof(pgfault))
        == 0)
        ;
    // 2. Handle IOPF if needed
    // 3. Return response
}
```

https://github.com/SamsungDS/libvfn/blob/iommufd-fault-queue/examples/iopf.c

USING CODE!

```
{
    // 1. Poll IOPF File Descriptor
    // 2. Handle IOPF if needed
    struct iommu_ioas_map map = {
        .size = sizeof(map),
        .flags = IOMMU_IOAS_MAP_READABLE |
        ...,
        .ioas_id = ioas->id,
        .user_va = (uint64_t)vaddr,
        .length = len,
    };
    ioctl(__iommufd, IOMMU_IOAS_MAP,
        &map);
    // 3. Return response
```

https://github.com/SamsungDS/libvfn/blob/iommufd-fault-queue/examples/iopf.c

USING CODE!

```
{
    // 1. Poll IOPF File Descriptor
    // 2. Handle IOPF if needed
    // 3. Return response
    struct iommu_hwpt_page_response pgresp
        = {
        .code = IOMMUFD_PAGE_RESP_SUCCESS,
        .cookie = pgfault.cookie;
    };
    write(fq.fault_fd, &pgresp,
        sizeof(pgresp));
}
```

https://github.com/SamsungDS/libvfn/blob/iommufd-fault-queue/examples/iopf.c

FORWARD?

- Relevant AMD, ARM, RISC-V?
- libvfn for iommu testing?