VIRTIO Introduction

--based on virtio-blk implementation

John.Gong

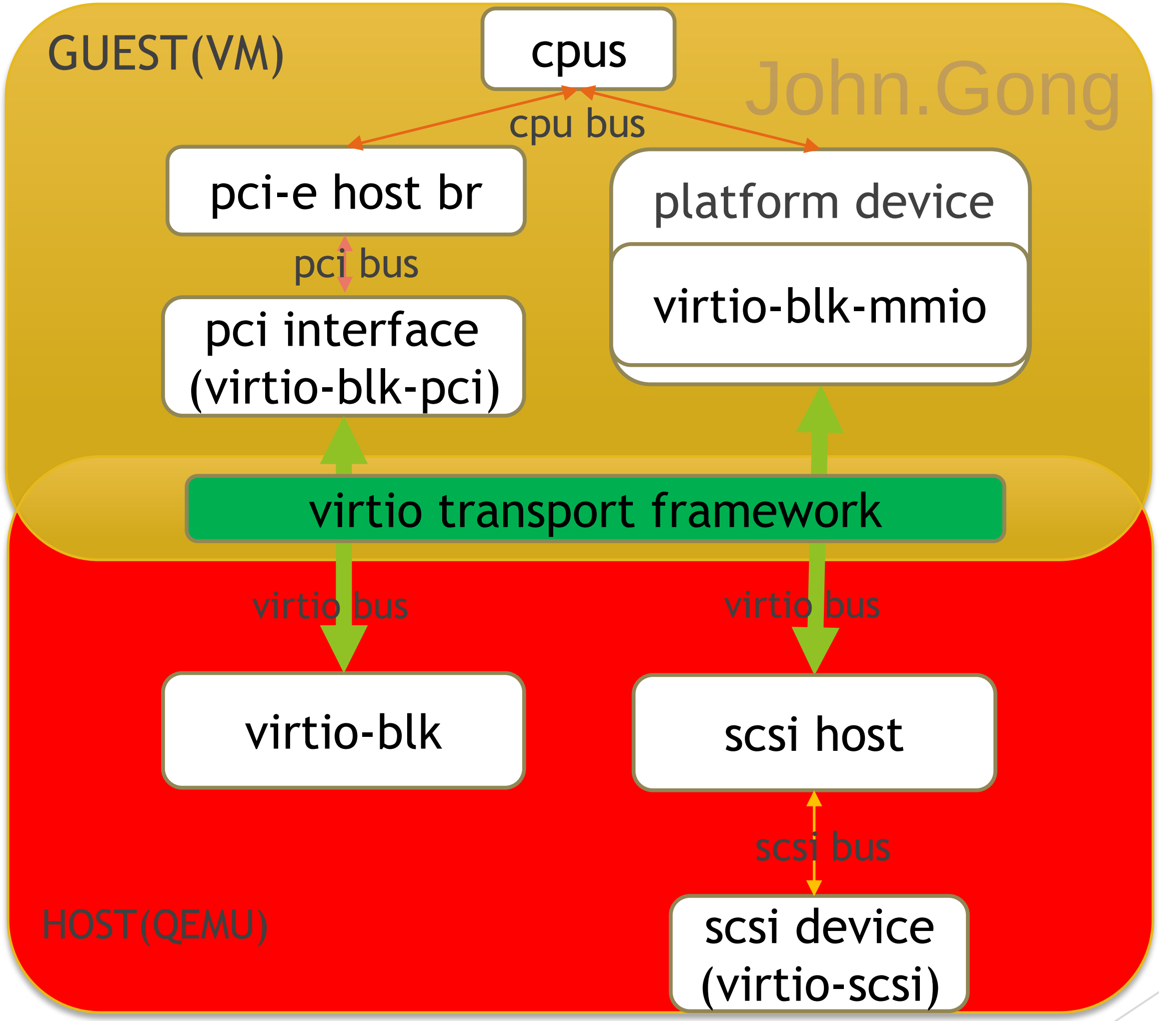
Agenda

- What is VIRTIO?
- Hierarchy
- Vring
- Vring management
- Guest read process
- Q & A
- Reference

What is VIRTIO

- VIRTIO is a virtual transport protocol, only exists in the para-virtualization environment. High performance due to:
  - share memory: no memory copy
  - lock-free queue: host and guest handle the queue concurrently

Hierarchy



GUEST(VM)

cpus

John.Gong

cpu bus

pci-e host br

platform device

pci bus

virtio-blk-mmio

pci interface (virtio-blk-pci)

virtio transport framework

virtio bus

virtio bus

virtio-blk

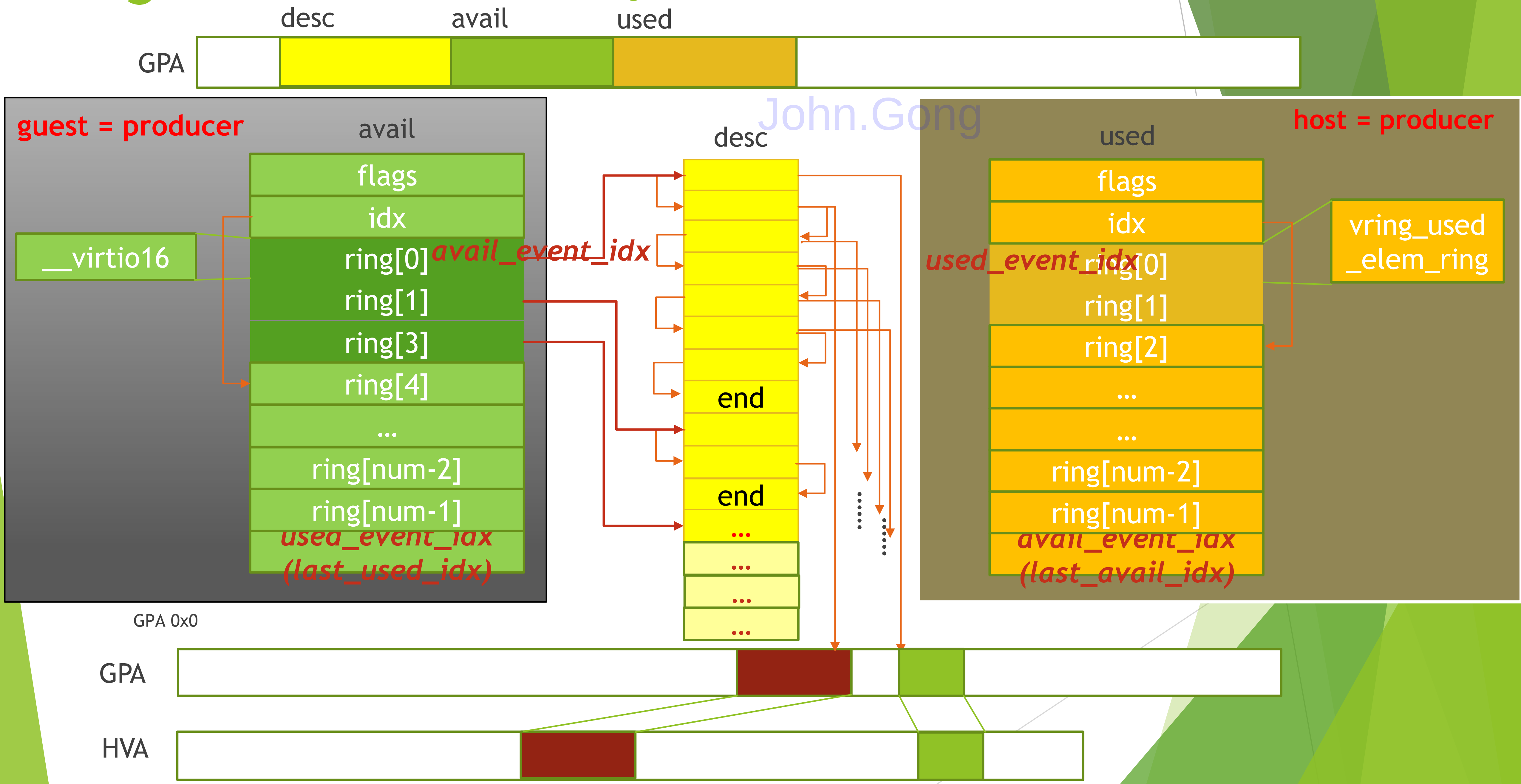
scsi host

scsi bus

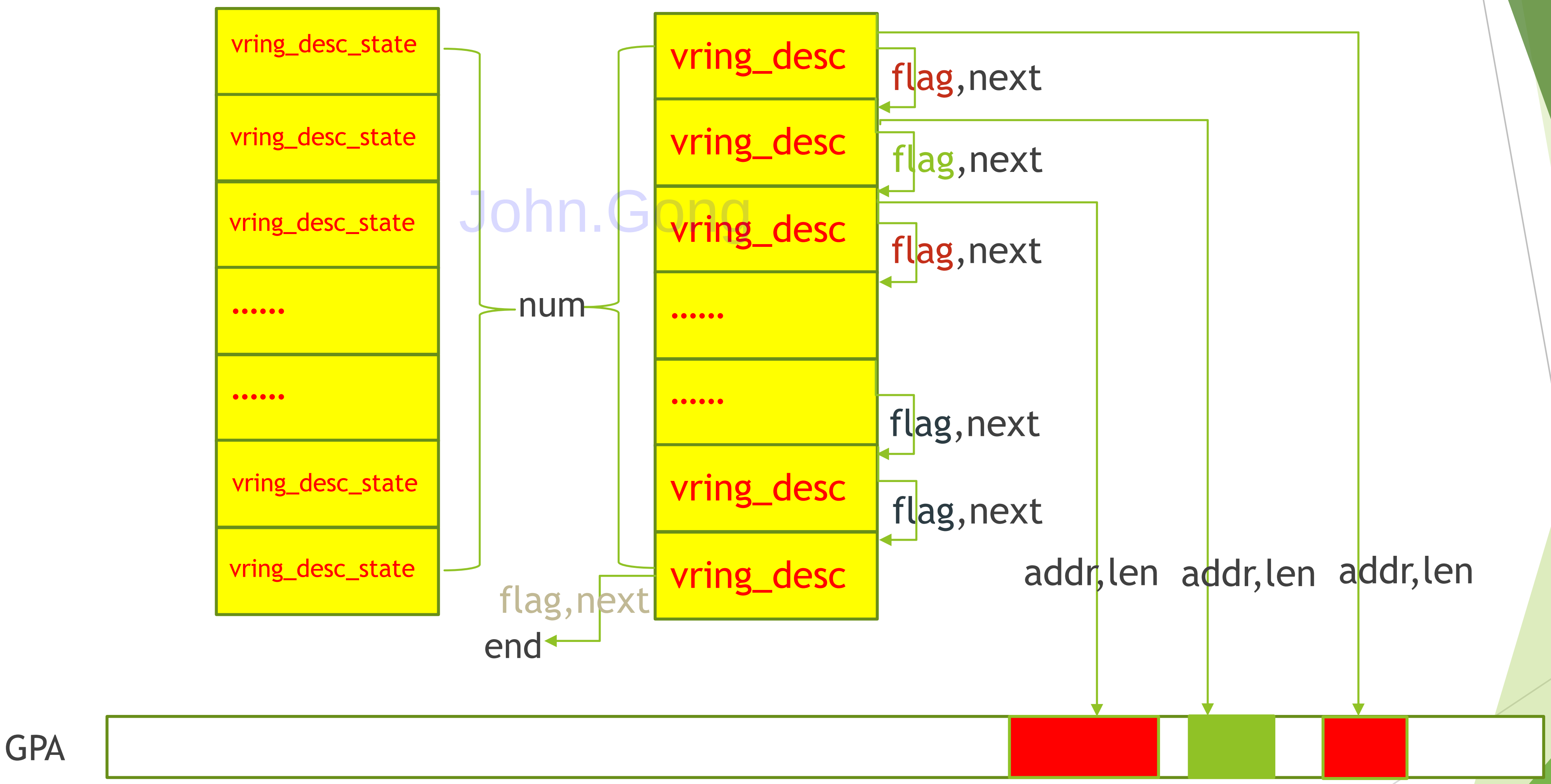
HOST(QEMU)

scsi device (virtio-scsi)

vring – share between host and guest



vring – vring_desc && vring_desc_state



Vring management – share with host and guest

```
struct vring_desc {
        __virtio64 addr;  //GPA
        __virtio32 len;
        __virtio16 flags;
        __virtio16 next;
};
struct vring_avail {
        __virtio16 flags;
        __virtio16 idx;
        __virtio16 ring[];
};
struct vring_used {
        __virtio16 flags;
        __virtio16 idx;
        struct vring_used_elem ring[];
};
```

```
struct vring_used_elem {
        __virtio32 id;
        __virtio32 len; //in or out data lens in byte
};
```

Vring management – guest private

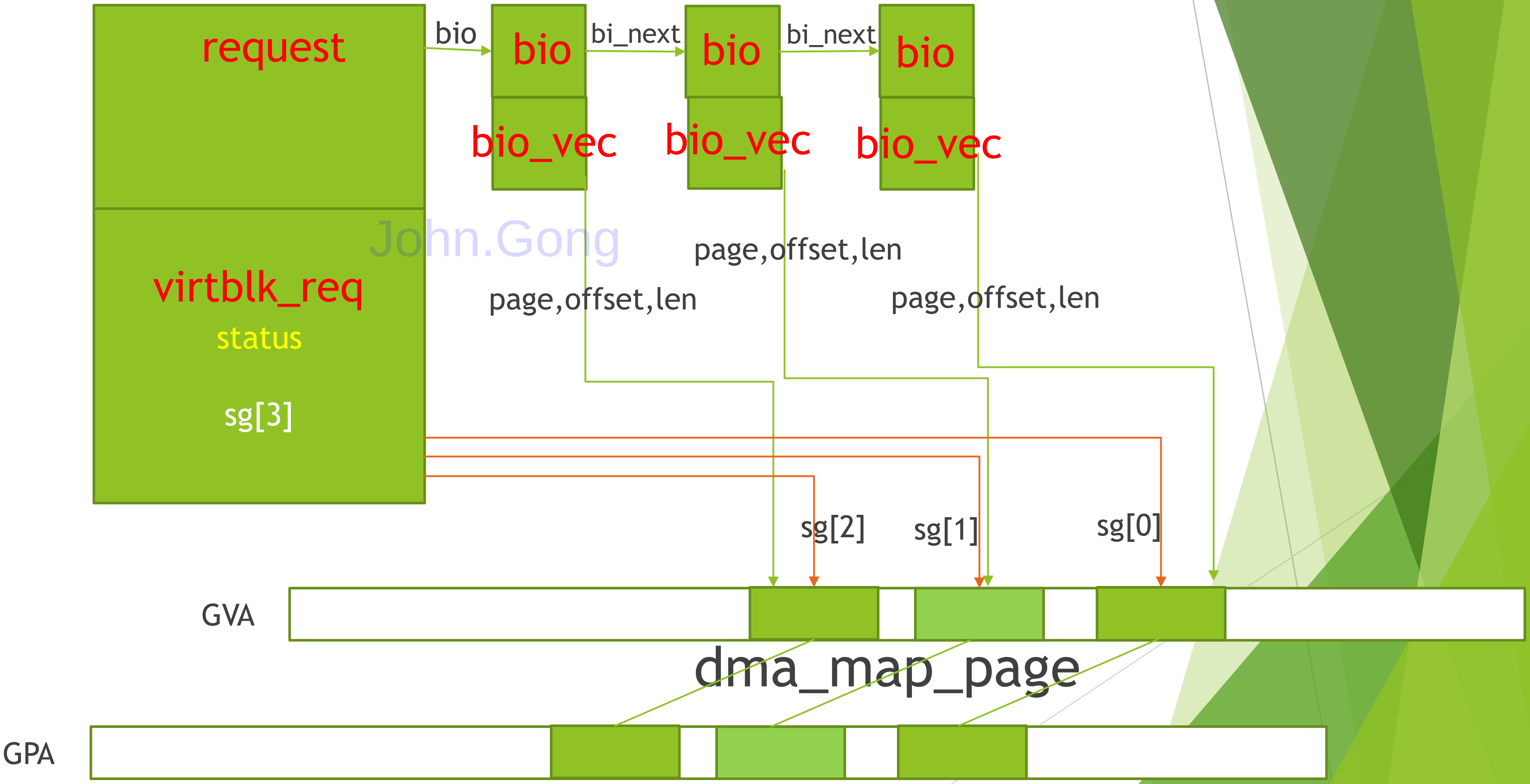
```
struct vring_virtqueue {
        struct virtqueue vq;
        struct vring vring;
        unsigned int free_head; //index to desc
        unsigned int num_added;
        u16 last_used_idx;//ci of the used ring, equal to used_event_idx
        u16 avail_flags_shadow;
        u16 avail_idx_shadow;
        bool (*notify)(struct virtqueue *vq);
        size_t queue_size_in_bytes;
        dma_addr_t queue_dma_addr;
        /* Per-descriptor state. */
        struct vring_desc_state desc_state[];
};
```

```
struct virtqueue {
        void (*callback)(struct virtqueue *vq);
        unsigned int index;
        unsigned int num_free;
};
struct vring {
        unsigned int num;
        struct vring_desc *desc;
        struct vring_avail *avail;
        struct vring_used *used;
};
struct vring_desc_state {
        void *data;
        struct vring_desc *indir_desc;
};
```

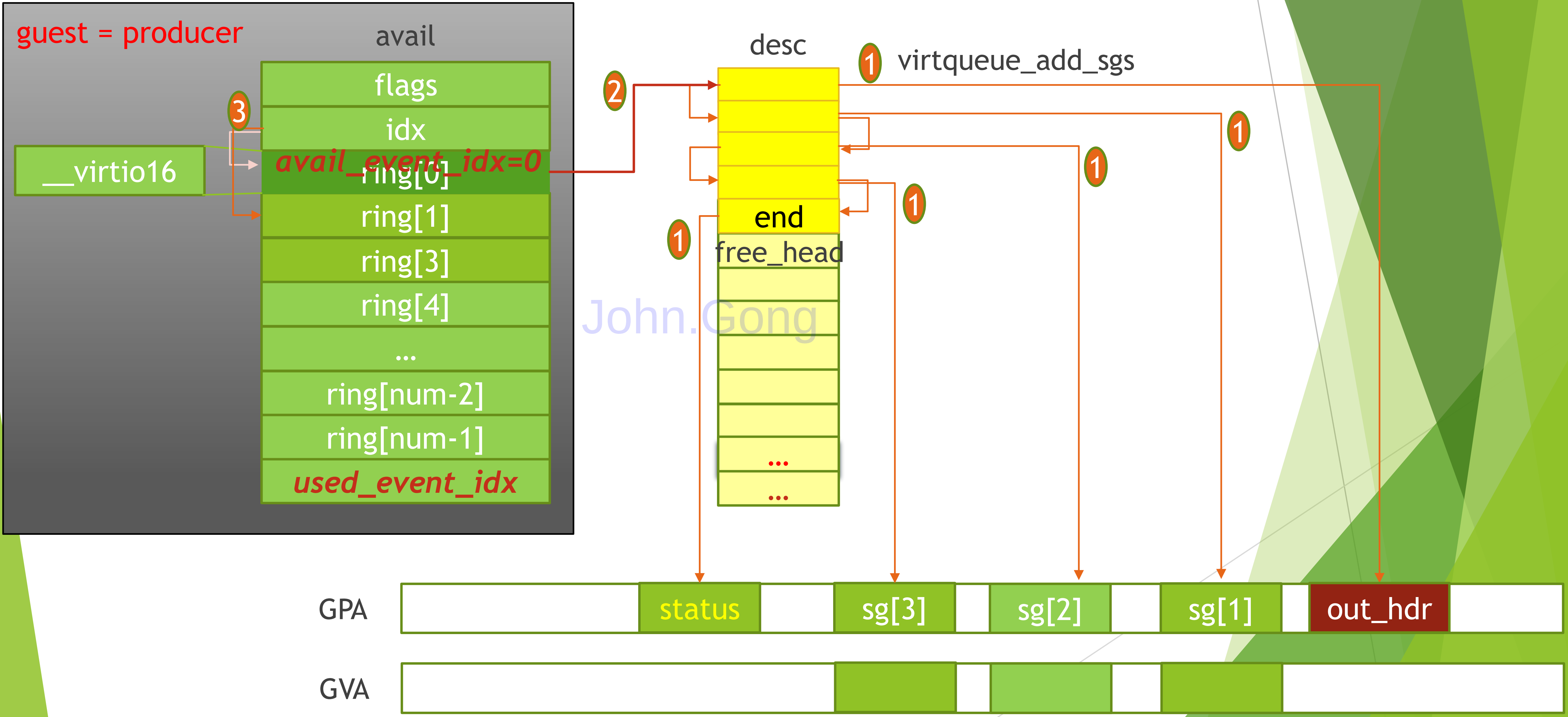
Vring management – host private

```
struct VirtQueue
{
    VRing vring;
    /* Next head to pop */
    uint16_t last_avail_idx;
    /* Last avail_idx read from VQ. */
    uint16_t shadow_avail_idx;
    uint16_t used_idx;
    uint16_t queue_index;
    unsigned int inuse;
    uint16_t vector;
    VirtIOHandleOutput handle_output;
};
```

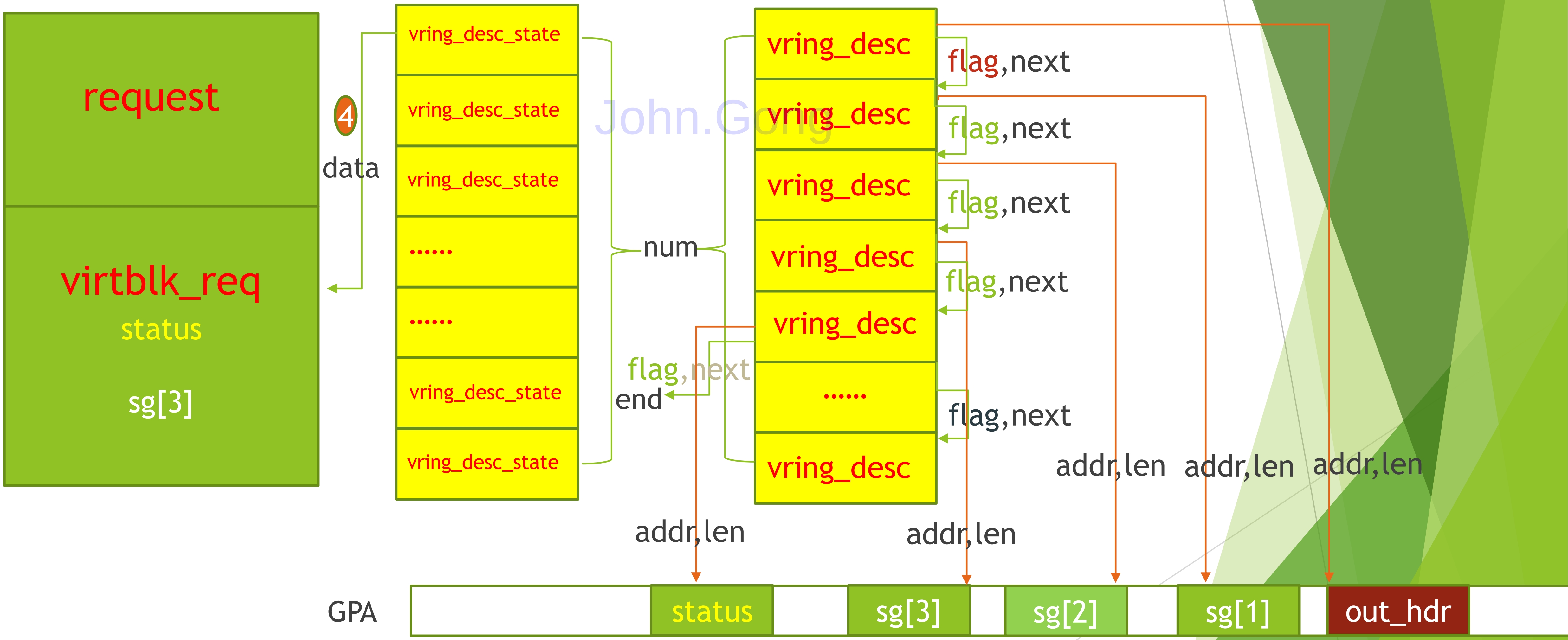
Guest read process (1)guest: virtblk_request



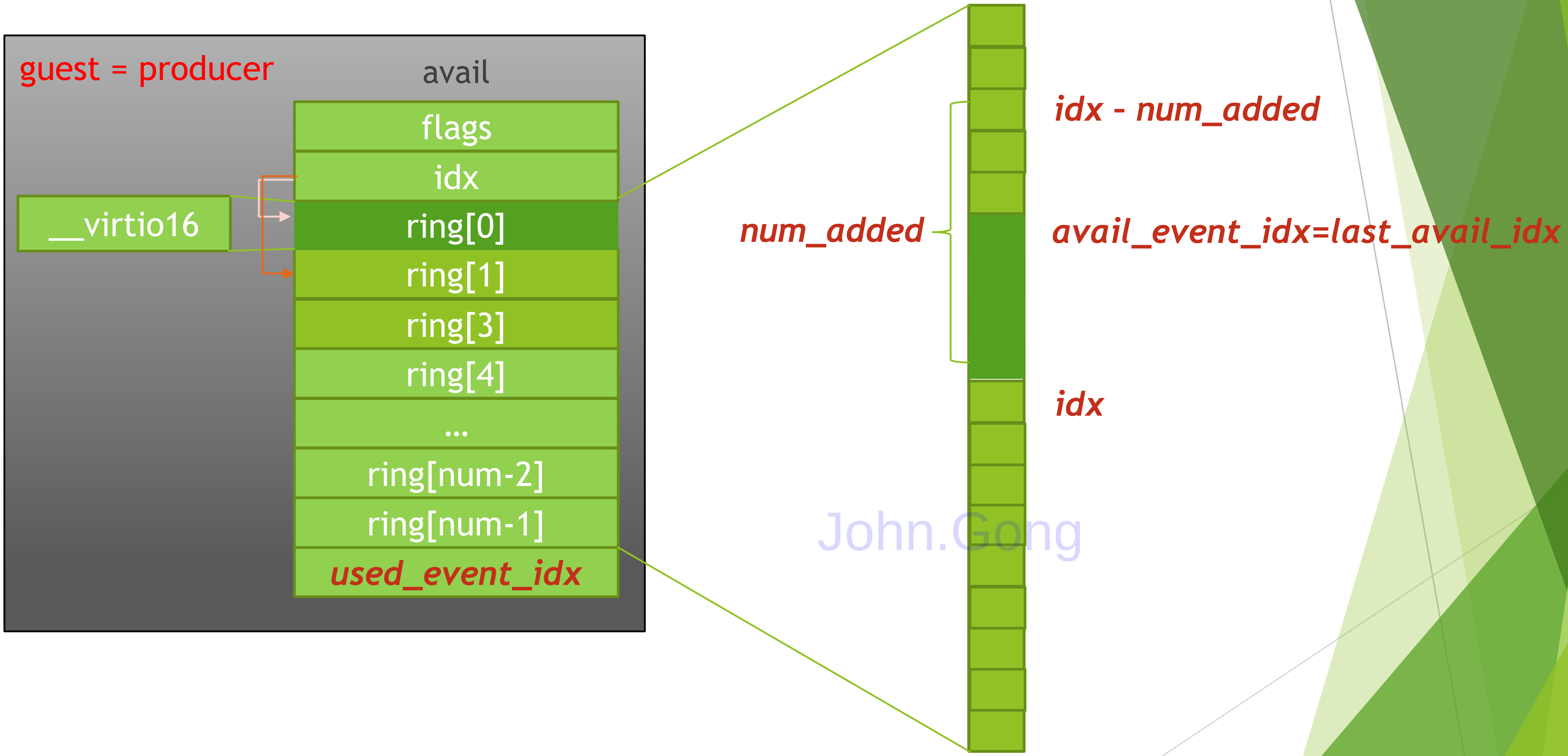
Guest read process (2)guest: virtqueue_add_sgs



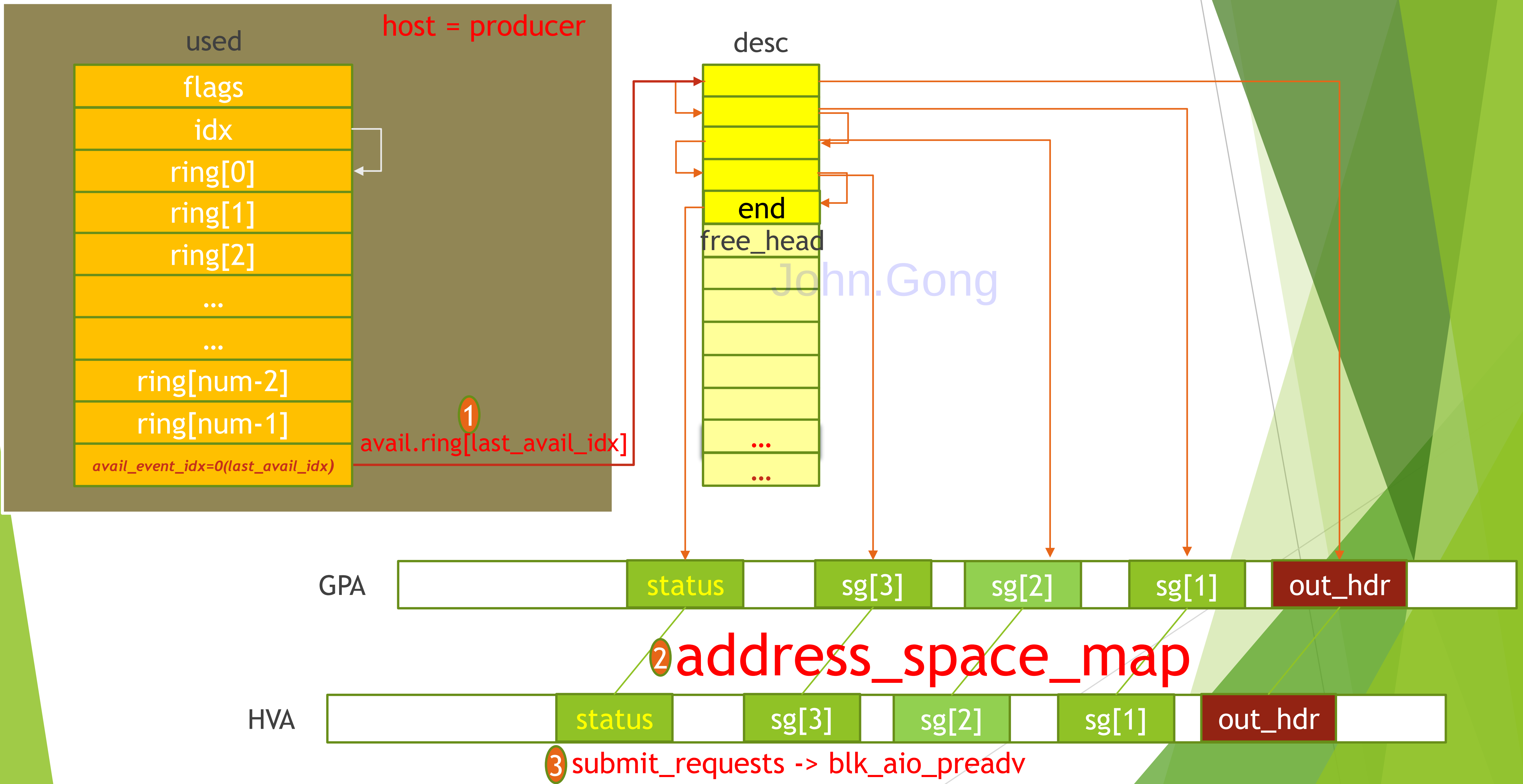
Guest read process (3)guest: save the virtblk_req to vring_desc_state



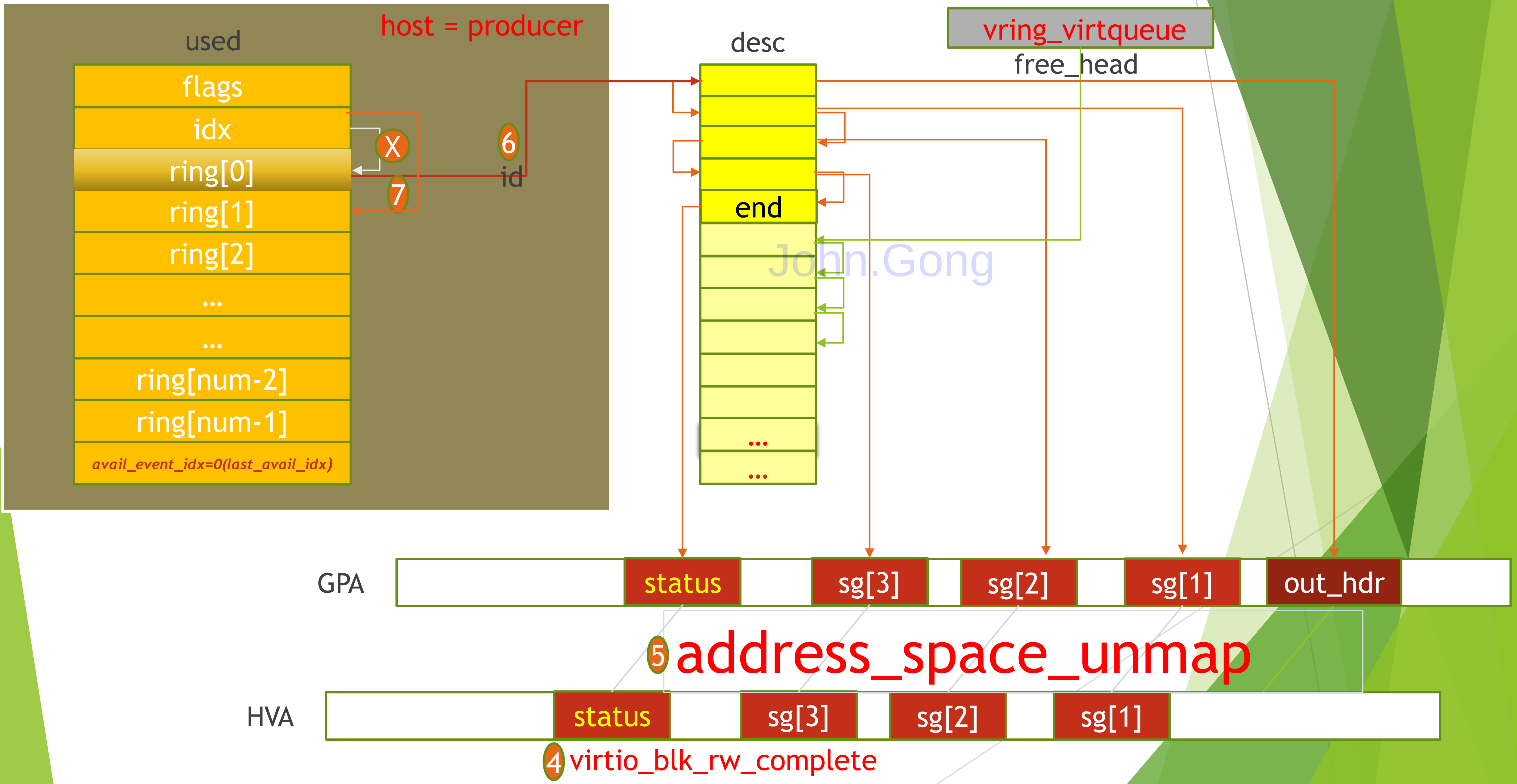
Guest read process (4)guest: virtqueue_kick_prepare && vp_notify



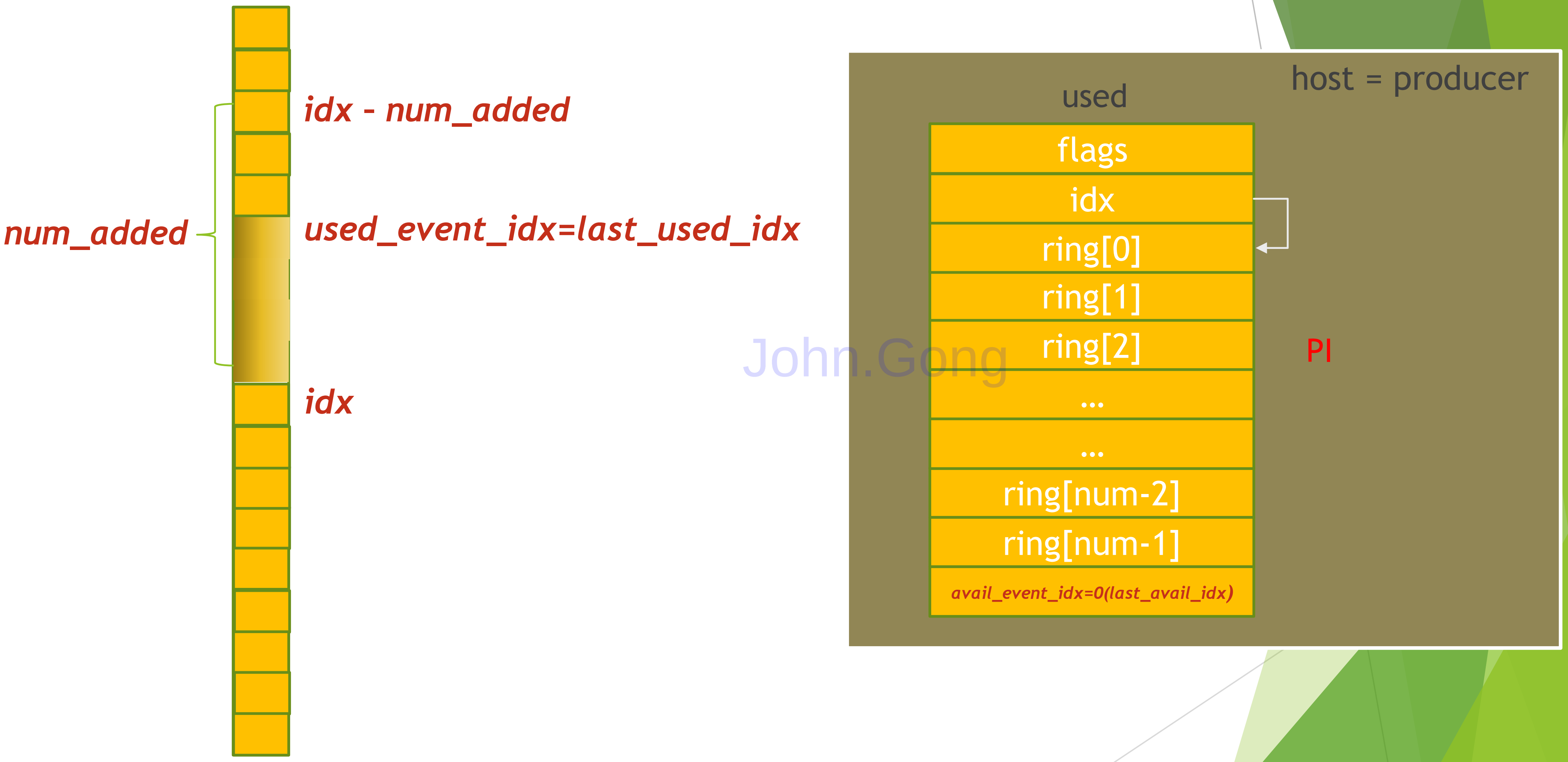
Guest read process (5)host: virtqueue_pop



Guest read process (6)host: virtqueue_push



Guest read process (7)host: virtio_notify



idx - num_added

num_added

used_event_idx=last_used_idx

idx

used

host = producer

flags

idx

ring[0]

ring[1]

ring[2]

…

…

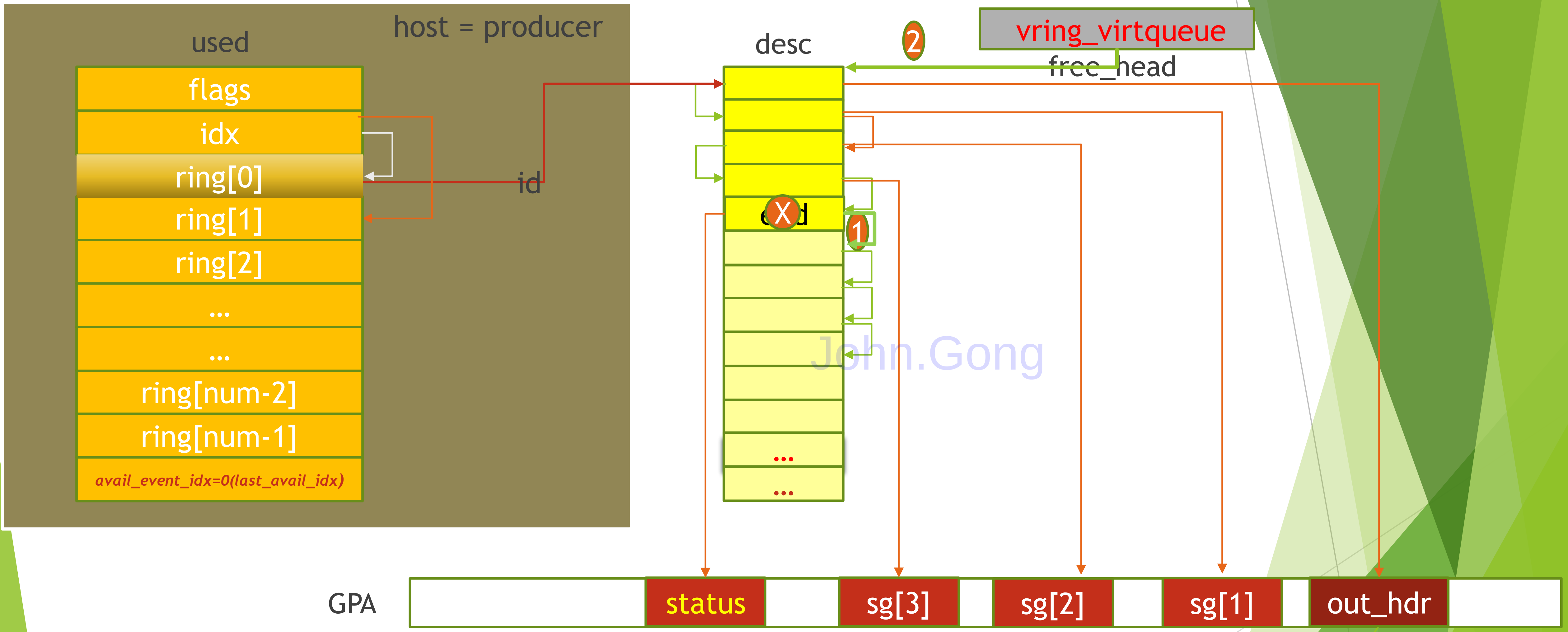
ring[num-2]

ring[num-1]

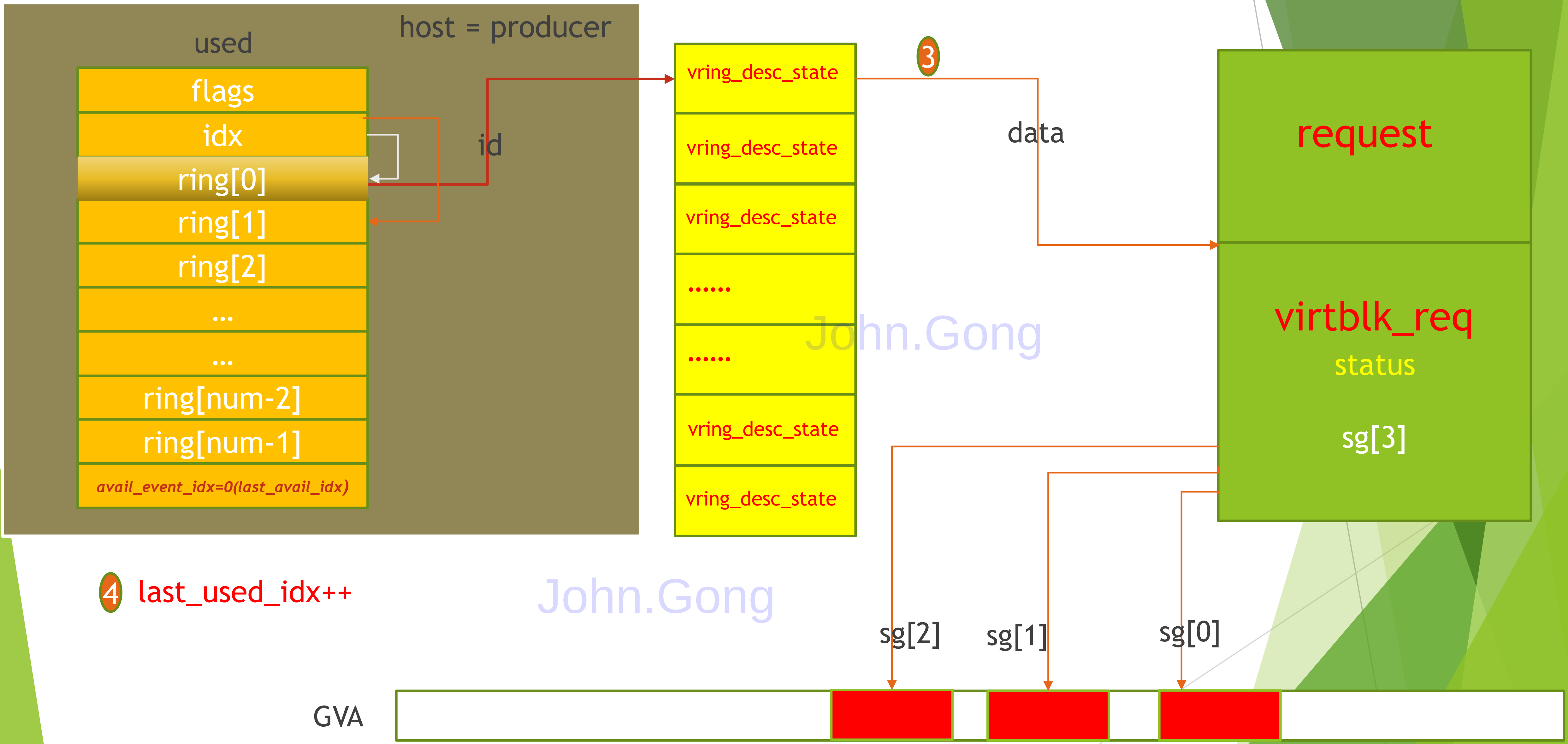
avail_event_idx=0(last_avail_idx)

PI

Guest read process (8)guest: vring_interrupt -> virtblk_done



Guest read process (9)guest: vring_interrupt -> virtblk_done



4 last_used_idx++

QA Discussion

- How to share the memory between the host(QEMU) and guest(VM) within the virtio implementation?
  - No extra action is needed to implement it. Host(QEMU) and guest(VM) are within the same process user space. Host(QEMU) and Guest(VM) can access the same HPA with the same HVA.

Reference

- Source code
  - linux kernel CN: 328b4ed93b69a6f2083d52f31a240a09e5de386a
  - qemu CN: eaefea537b476cb853e2edbdc68e969ec777e4bb

Foxit Reader PDF Document

Foxit Reader PDF Document

Foxit Reader PDF Document

Foxit Reader PDF Document

Thank You !

John.Gong