本文档介绍了多队列（queues）网卡DMA接收数据包方式：

1. Queues：增加驱动I/O能力，提供多个输入队列
2. DMA：减少CPU负载处理

一． 主要数据结构

Receive descriptor node

struct rxbd\_desc{

u16 status; /\* Status Fields\*/ /\* optional \*/

u16 length; /\* Buffer Length \*/ /\* optional \*/

u32 bufPtr; /\* Buffer Pointer 通常DMA指向skb->data\*/

};

Rx\_buffer:skb

struct rx\_buffer {

struct sk\_buff \*skb;

………

};

Queue ring descriptor: ring->desc包含count个struct rxbd\_desc和struct rx\_buffer

Struct ring {

void \*desc; /\* descriptor ring memory \*/

struct rx\_buffer \*rx\_buffer\_info; /\*skb data\*/

u16 count; /\* amount of descriptors \*/

unsigned int size; /\* length in bytes (of descriptors) \*/

dma\_addr\_t dma; /\* phys. address of descriptor ring \*/

next\_to\_clean；

next\_to\_use；

……..

}

二．主要寄存器：每个queue都有下列一组寄存器, n表示queue编号

1. **Receive Descriptor Base Address(RDBA[n])**：存放ring->desc的dma地址

2. **Receive Descriptor Length（RDLEN[n]）**: 存放该ring的描述符长度（ring->count \* sizeof(struct rxdb\_desc)）

3. **Receive Descriptor Control（RXDCTL[n]）**：queue使能控制器

三.程序分析

1. 在驱动**probe函数**中为每个queue分配一个ring：

static int alloc\_rx\_queues(struct ixgbe\_adapter \*adapter)

{

int rx = 0, nid = adapter->node;

for (; rx < adapter->num\_rx\_queues; rx++) {

struct ring \*ring;

ring = kzalloc\_node(sizeof(\*ring), GFP\_KERNEL, nid);

if (!ring)

ring = kzalloc(sizeof(\*ring), GFP\_KERNEL);

if (!ring)

goto err\_allocation;

ring->count = adapter->rx\_ring\_count;

ring->queue\_index = rx;

ring->reg\_idx = ring->queue\_index； /\*用于I/O寄存器编号绑定 通常顺序绑定 可自定义\*/

……..

…….

adapter->rx\_ring[rx] = ring;

}

return 0;

err\_allocation:

while (rx)

kfree(adapter->rx\_ring[--rx]);

return -ENOMEM;

}

1. 在**open函数**中为ring分配资源： 包括count个struct rx\_buffer和struct rxbd\_desc

int setup\_rx\_resources(struct ring \*rx\_ring)

{

int size;

size = sizeof(struct rx\_buffer) \* rx\_ring->count;

rx\_ring->rx\_buffer\_info = vzalloc\_node(size, rx\_ring->numa\_node);

if (!rx\_ring->rx\_buffer\_info)

rx\_ring->rx\_buffer\_info = vzalloc(size);

if (!rx\_ring->rx\_buffer\_info)

goto err;

/\* Round up to nearest 4K \*/

rx\_ring->size = rx\_ring->count \* sizeof(struct rxbd\_desc);

rx\_ring->size = ALIGN(rx\_ring->size, 4096);

rx\_ring->desc = dma\_alloc\_coherent(dev, rx\_ring->size,

&rx\_ring->dma, GFP\_KERNEL);

/\* dma\_alloc\_coherent 分配rx\_ring->size 大小的dma内存，虚拟地址给描述符desc，物理地址给rx\_ring->dma，rx\_ring->dma值将被填入**RDBA**寄存器 \*/

if (!rx\_ring->desc)

goto err;

rx\_ring->next\_to\_clean = 0;

rx\_ring->next\_to\_use = 0;

return 0;

err:

vfree(rx\_ring->rx\_buffer\_info);

rx\_ring->rx\_buffer\_info = NULL;

return -ENOMEM;

}

1. 在**open函数**中为将ring与receive describtor 寄存器相绑定：

void configure\_rx\_ring(struct adapter \*adapter,

struct ring \*ring)

{

struct hw \*hw = &adapter->hw;

u64 rdba = ring->dma;

u32 rxdctl;

u8 reg\_idx = ring->reg\_idx;/\*ring 与寄存器组对应的编号，表示为该编号的I/Oqueue\*/

/\* disable queue to avoid issues while updating state \*/

rxdctl = IXGBE\_READ\_REG(hw, RXDCTL(reg\_idx));

ixgbe\_disable\_rx\_queue(adapter, ring);

/\*设置RXDCTL[ring->reg\_idx] 关闭queue使能\*/

IXGBE\_WRITE\_REG(hw, RDBA(reg\_idx), rdba )

/\*设置RDBA [ring->reg\_idx] 为 ring->dma \*/

IXGBE\_WRITE\_REG(hw, RDLEN(reg\_idx),

ring->count \* sizeof(struct\_rxdb\_desc));

/\*设置RDLEN [ring->reg\_idx] 为count\*desc\_len \*/

/\* enable receive descriptor ring \*/

rxdctl |= IXGBE\_RXDCTL\_ENABLE;

IXGBE\_WRITE\_REG(hw, RXDCTL(reg\_idx), rxdctl);

}

1. 在**open函数**中为ring的count个desc-> bufPtr分配空间,进行DMA绑定skb->data：

alloc\_rx\_buffers(struct ring \*rx\_ring, u16 cleaned\_count);

{

struct rxdb\_desc \*rx\_desc;

struct rx\_buffer \*bi;

struct sk\_buff \*skb;

u16 i = rx\_ring->next\_to\_use;

while (cleaned\_count--) {

rx\_desc = &(struct rxdb\_desc \*)(( rx\_ring)->desc))[i]) /\*取第i个描述符\*/

bi = &rx\_ring->rx\_buffer\_info[i]; /\*取第i个buffer\*/

skb = bi->skb;

/\*分配skb\*/

skb = netdev\_alloc\_skb\_ip\_align(rx\_ring->netdev, rx\_ring->rx\_buf\_len);

bi->skb = skb;

/\*skb->data 映射给描述符bufPtr\*/

rx\_desc->bufPtr = dma\_map\_single(rx\_ring->dev,skb->data,

rx\_ring->rx\_buf\_len, DMA\_FROM\_DEVICE);

}

}