QoS Implementation with DPDK

515030910223 杨健邦

使用的DPDK API

1. Meter

```
int
rte_meter_srtcm_config(struct rte_meter_srtcm *m,
  struct rte_meter_srtcm_params *params);
```

- 初始化每个流的srtcm的runtime data，srtcm每个流一个。

```
static inline enum rte_meter_color
rte_meter_srtcm_color_blind_check(struct rte_meter_srtcm *m,
  uint64_t time,
  uint32_t pkt_len);
```

- 给到来的每一个包染色，传入的srtcm为每个流对应的runtime data。**注意: 此处的time是以cycle为单位的，而且在上面config的时候会调用api来记录config的时间，因此这里的time并不是从0开始的，需要加一个偏移量。**

2. Dropper

```
int
rte_red_config_init(struct rte_red_config *red_cfg,
   const uint16_t wq_log2,
   const uint16_t min_th,
   const uint16_t max_th,
   const uint16_t maxp_inv);
```

- 初始化config， config每个流的每一种颜色都要一个， 一共4 * 3 = 12个。

```
int
rte_red_rt_data_init(struct rte_red *red);
```

- 初始化dropper的runtime data， 也是12个。

```
 static inline int
 rte_red_enqueue(const struct rte_red_config *red_cfg,
    struct rte_red *red,
    const unsigned q,
    const uint64_t time);
```

- 对于每一个到来的包， 判断其是否需要被丢弃。 **注意: 这里的q为每一个流一个(DPDK上面文档有说明)， 单位为packets， 每一个period(1,000,000个cycles)清空一次。**

参数理解

- Meter

```
/* srtcm 每个参数的说明以及作用:
 *  cir:
```

```
 *       - 承诺访问速率,每秒钟往C桶和E桶填充新令牌的速率,一个令牌相当于一个Byte
 *       - 单位 Byte/s
 *  cbs:
 *     - C桶容量
 *     - 增大的话,承受burst的能力增强
 *     - 单位 Byte
 *  ebs:
 *     - E桶容量
 *     - 增大的话,承受burst的能力增强
 *     - 单位 Byte
 */
```

- Dropper

```
/*
 * WRED 每个参数的说明以及作用:
 *  min_th:
 *      - 小队列长度,当队列小于该长度时,不会丢包,在min和max之间开始丢包，丢包可能性随q增大而增大，最大丢包可能性为maxp
 *  max_th:
 *      - 最大队列长度,当队列大于该长度时,丢包率为100%
 *  maxp_inv:
 *      - 队列长度在min和max之间时最大的丢包可能性, 10表示, 10个包中有1个包会丢
 *  wq_log2:
 *      - 决定平均队列长度变化速率的快慢,同一种流的wq_log2的值要相同。
 */
```

Meter的调参过程

1. 通过调用rte_get_tsc_hz()， 可知道虚拟机CPU的HZ为3,095,221,586, 也就是说每秒中CPU运行3,095,221,586个cycle。
2. meter将cir转换为cir_period和cir bytes per period， cir_period指的是每隔多少个cycles填充一次令牌桶，CIR bytes per period 指的是每个period填充多少个bytes。
3. 通过计算main中发包速率，得出每隔1,000,000个cycles,平均每一个流要发(1000/4)Packets * 640 Bytes = 160,000, 即每秒每个流要发送495,235,453.76Bytes
4. 对于FLOW 0，要让其获得最大带宽，则其可能的最大发包速率为(128+1024)*1500 = 1,728,000 Byte， cbs和ebs应该设得尽可能大，使其的包都染成绿色。
5. 对于FLOW 1，其cir应为FLOW 0的一半， 调整cbs和ebs， 使得FLOW 1中被染成绿包的数量约等于FLOW 0绿包数量的一半偏少， 黄包的数量和绿包差不多， 其它都为红包。
6. 同理， 其它流的设置也类似，逐次减半。
7. **最后的结果是: 由于FLOW 0要达到最大带宽，所以FLOW 0全都是绿包，其它流的绿包数分别约为FLOW 0的绿包数的二分之一、四分之一、八分之一**

Dropper的调参过程

1. 由于FLOW 0可以得到最大带宽，所以其min_th和max_th要调得尽可能大，分别为1022和1023，而丢包率要尽可能低，因此将maxp_inv设置为255, 255个包才会丢一个。
2. 由于上面Meter的时候设置的绿包比差不多等于带宽比，其它Flow的绿包也要尽可能地少丢，但是也不能不丢，因此，将绿包的min_th设置为64左右，而将绿包的max_th设置为1023。
3. 由于上面Meter的时候设置的绿包比差不多等于带宽比，FLOW1-3的黄包和红包丢包率要较大，方法是将红包和黄包的min_th和max_th设置得比较小(1-24左右)，同时maxp_inv也设置得比较小(1-4左右)。
4. 剩下的工作就是细调, 使得四个流的带宽比为8:4:2:1

实际结果

- 平均每个流一共要发送1,600,000个Bytes，因此FLOW 0全部发送，不丢包，而FLOW 1约能发送800,000个Bytes, FLOW 2约能发送400,000个Bytes，FLOW 3 约能发送200,000个Bytes。

- 测试1

```
QoS Menter: hz = 3095224749
METER: Low level srTCM config:
        CIR period = 105, CIR bytes per period = 17
METER: Low level srTCM config:
        CIR period = 111, CIR bytes per period = 9
METER: Low level srTCM config:
        CIR period = 124, CIR bytes per period = 5
METER: Low level srTCM config:
        CIR period = 149, CIR bytes per period = 3
fid: 0, send: 1792000, pass: 1792000, green: 2758, yellow: 72, red: 0
fid: 1, send: 1765744, pass: 815481, green: 1122, yellow: 716, red: 928
fid: 2, send: 1843853, pass: 420477, green: 492, yellow: 415, red: 1944
fid: 3, send: 1800672, pass: 211493, green: 266, yellow: 324, red: 2264
```

- 测试2

```
QoS Menter: hz = 3095217724
METER: Low level srTCM config:
        CIR period = 105, CIR bytes per period = 17
METER: Low level srTCM config:
        CIR period = 111, CIR bytes per period = 9
METER: Low level srTCM config:
        CIR period = 124, CIR bytes per period = 5
METER: Low level srTCM config:
        CIR period = 149, CIR bytes per period = 3
fid: 0, send: 1527701, pass: 1527701, green: 2391, yellow: 0, red: 0
fid: 1, send: 1569882, pass: 801297, green: 1089, yellow: 713, red: 617
fid: 2, send: 1490874, pass: 411072, green: 504, yellow: 414, red: 1421
fid: 3, send: 1497624, pass: 211881, green: 266, yellow: 328, red: 1755
```

- 测试3

```
QoS Menter: hz = 3095219664
METER: Low level srTCM config:
        CIR period = 105, CIR bytes per period = 17
METER: Low level srTCM config:
        CIR period = 111, CIR bytes per period = 9
METER: Low level srTCM config:
        CIR period = 124, CIR bytes per period = 5
METER: Low level srTCM config:
        CIR period = 149, CIR bytes per period = 3
fid: 0, send: 1547511, pass: 1547511, green: 2422, yellow: 0, red: 0
fid: 1, send: 1572149, pass: 798046, green: 1114, yellow: 729, red: 642
fid: 2, send: 1528031, pass: 405771, green: 503, yellow: 416, red: 1509
fid: 3, send: 1578932, pass: 199376, green: 253, yellow: 307, red: 1915
```

- 测试4

```
QoS Menter: hz = 3095219664
METER: Low level srTCM config:
        CIR period = 105, CIR bytes per period = 17
METER: Low level srTCM config:
        CIR period = 111, CIR bytes per period = 9
METER: Low level srTCM config:
        CIR period = 124, CIR bytes per period = 5
METER: Low level srTCM config:
        CIR period = 149, CIR bytes per period = 3
fid: 0, send: 1547511, pass: 1547511, green: 2422, yellow: 0, red: 0
fid: 1, send: 1572149, pass: 798046, green: 1114, yellow: 729, red: 642
fid: 2, send: 1528031, pass: 405771, green: 503, yellow: 416, red: 1509
fid: 3, send: 1578932, pass: 199376, green: 253, yellow: 307, red: 1915
```

- 测试5

```
QoS Menter: hz = 3095218598
METER: Low level srTCM config:
        CIR period = 105, CIR bytes per period = 17
METER: Low level srTCM config:
        CIR period = 111, CIR bytes per period = 9
METER: Low level srTCM config:
        CIR period = 124, CIR bytes per period = 5
METER: Low level srTCM config:
        CIR period = 149, CIR bytes per period = 3
fid: 0, send: 1468614, pass: 1468614, green: 2304, yellow: 0, red: 0
fid: 1, send: 1388155, pass: 799065, green: 1105, yellow: 730, red: 385
fid: 2, send: 1510013, pass: 413480, green: 529, yellow: 426, red: 1426
fid: 3, send: 1502140, pass: 212099, green: 258, yellow: 309, red: 1742
```