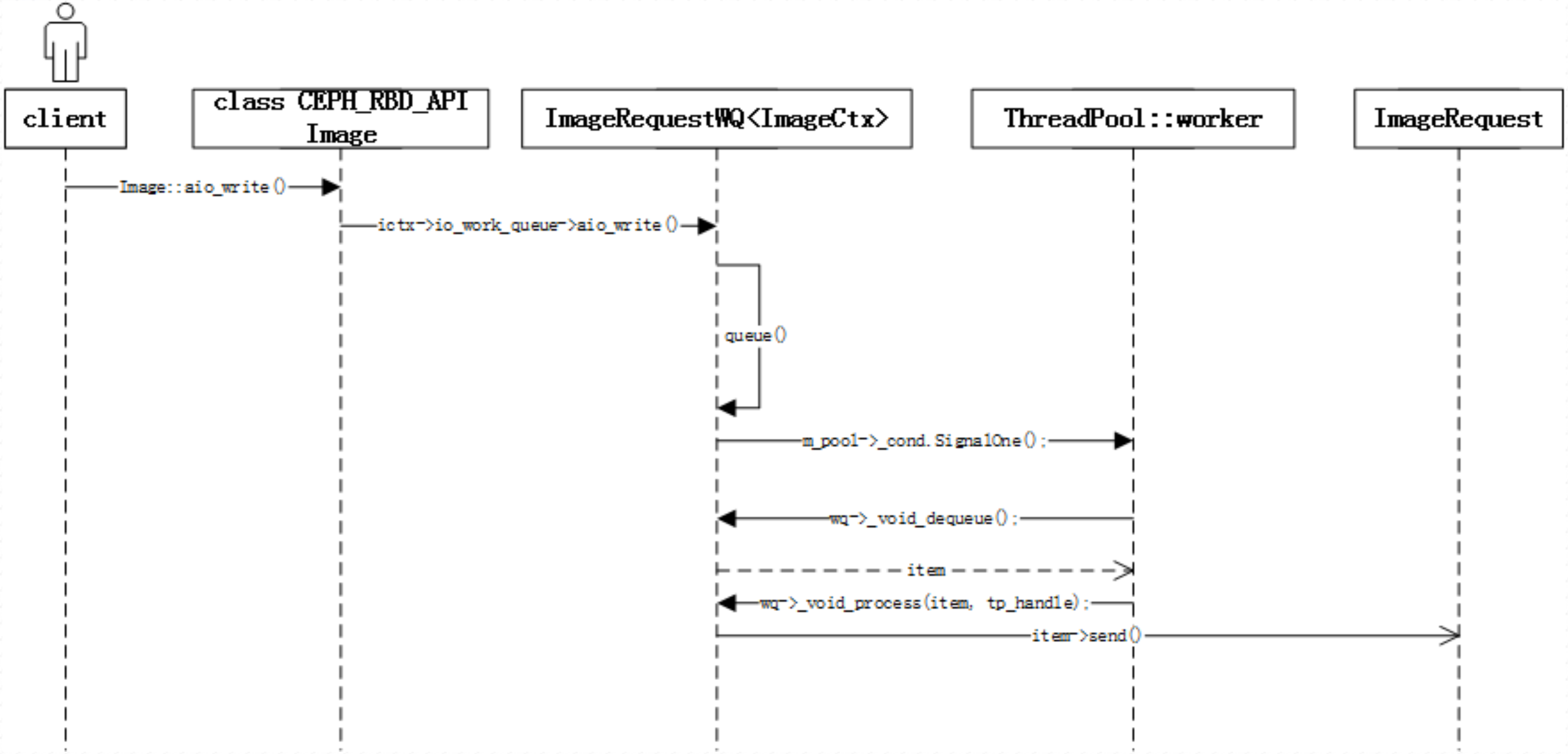
浅谈RBD 的QOS（Rate Limiting）

杨东升

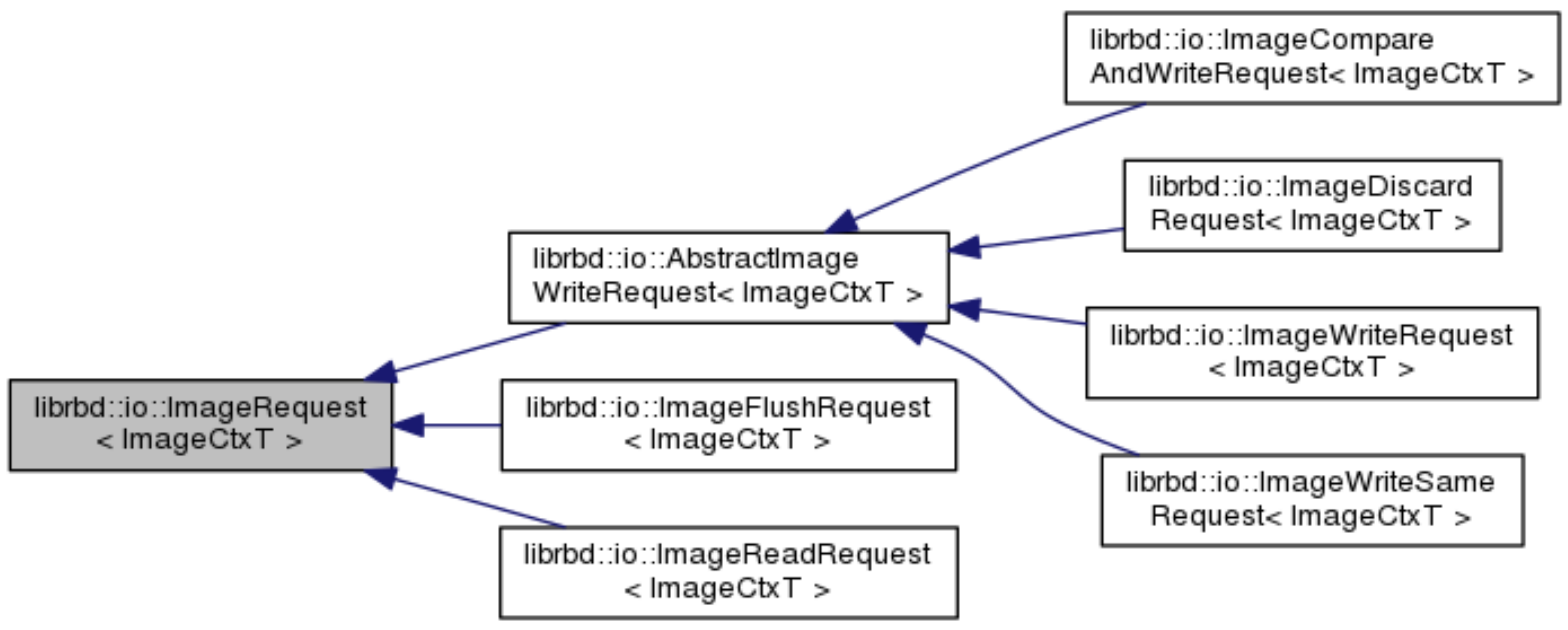
为什么需要RBD 的rate limiting

- 总的IO 能力是有限的
- 尽量保证IO能力的合理分配
- Cinder back-end QOS
- ISCSI , ndb …

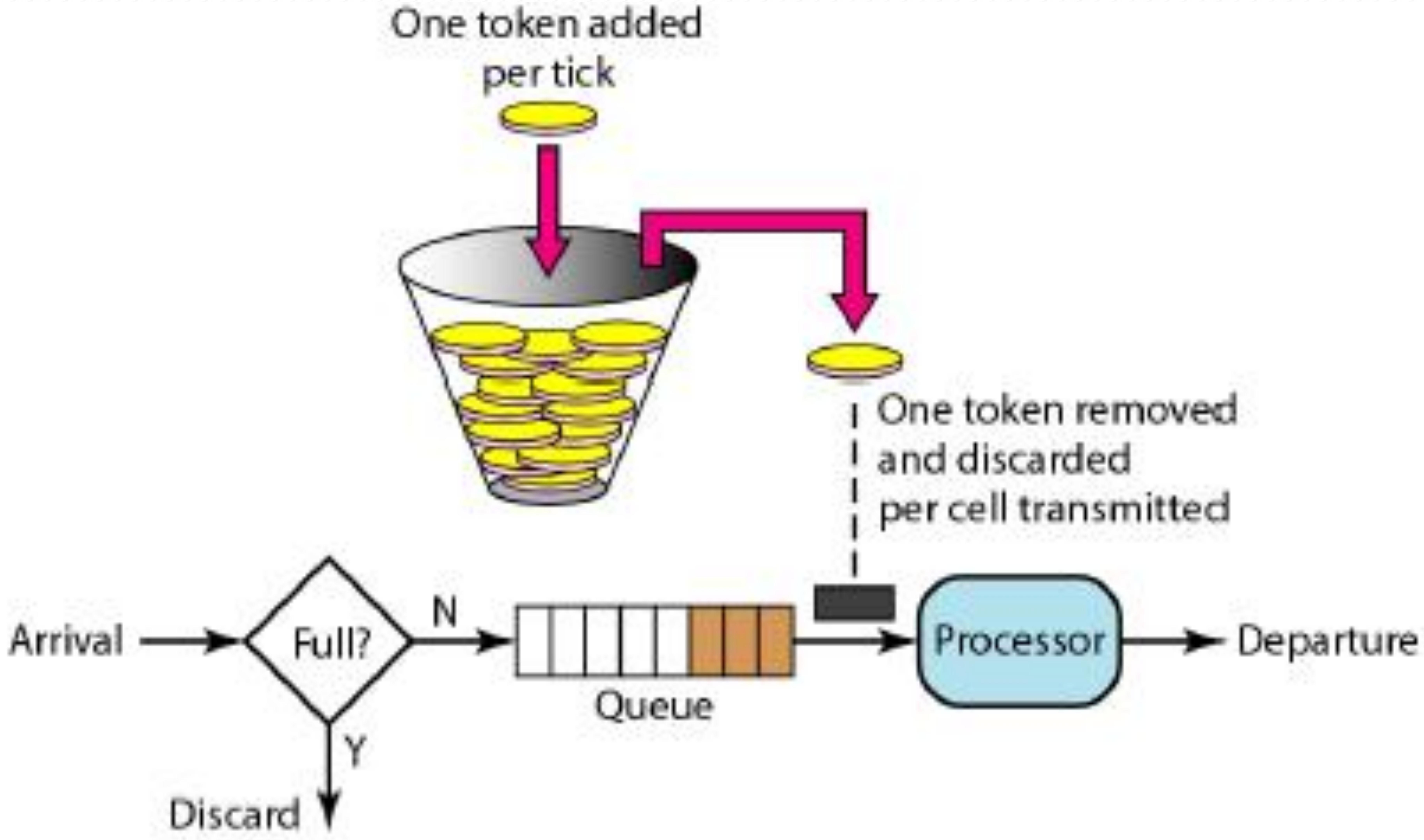
RBD 读写流程简介


client
class CEPH_RBD_API Image
ImageRequestWQ<ImageCtx>
ThreadPool::worker
ImageRequest
Image::aio_write()
ictx->io_work_queue->aio_write()
queue()
m_pool->_cond.SignalOne();
wq->_void_dequeue();
item
wq->_void_process(item, tp_handle);
item->send()

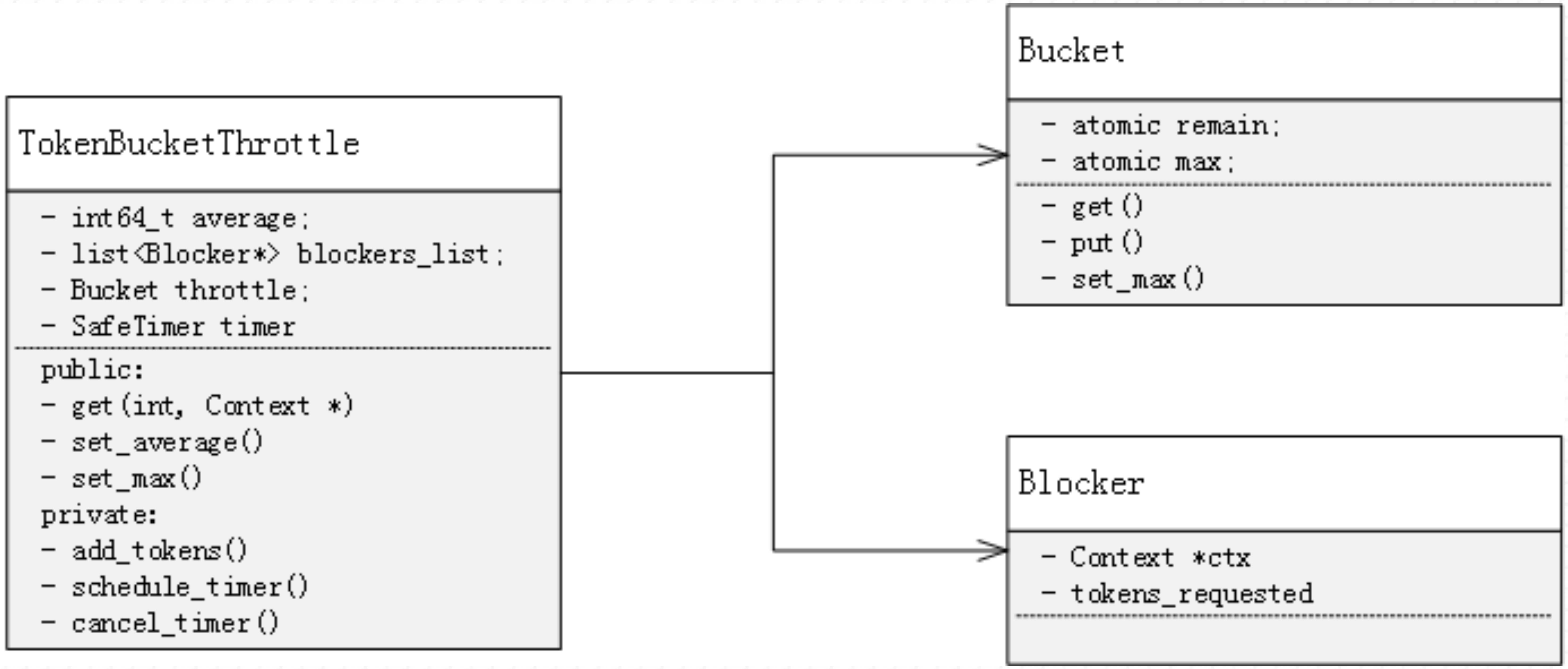
RBD 读写流程简介


librbd::io::ImageCompare
AndWriteRequest< ImageCtxT >
librbd::io::ImageDiscard
Request< ImageCtxT >
librbd::io::AbstractImage
WriteRequest< ImageCtxT >
librbd::io::ImageWriteRequest
< ImageCtxT >
librbd::io::ImageRequest
< ImageCtxT >
librbd::io::ImageFlushRequest
< ImageCtxT >
librbd::io::ImageWriteSame
Request< ImageCtxT >
librbd::io::ImageReadRequest
< ImageCtxT >

Tokenbucket 简介


One token added per tick
One token removed and discarded per cell transmitted
Arrival
Full?
N
Y
Discard
Queue
Processor
Departure

TokenBucketThrottle



TokenBucketThrottle

- int64_t average;
- list<Blocker*> blockers_list;
- Bucket throttle;
- SafeTimer timer

public:

- get(int, Context *)
- set_average()
- set_max()

private:

- add_tokens()
- schedule_timer()
- cancel_timer()

Bucket

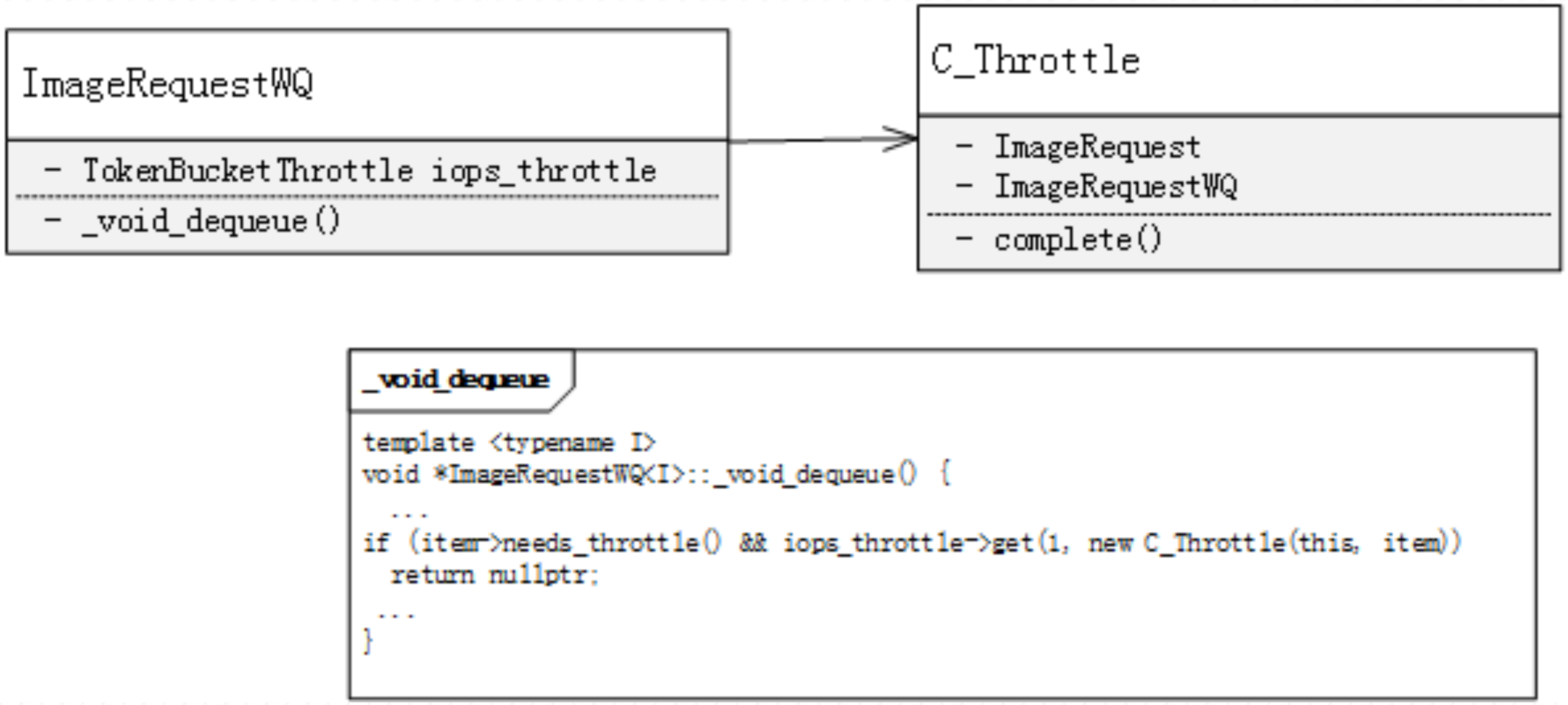
- atomic remain;
- atomic max;

- get()
- put()
- set_max()

Blocker

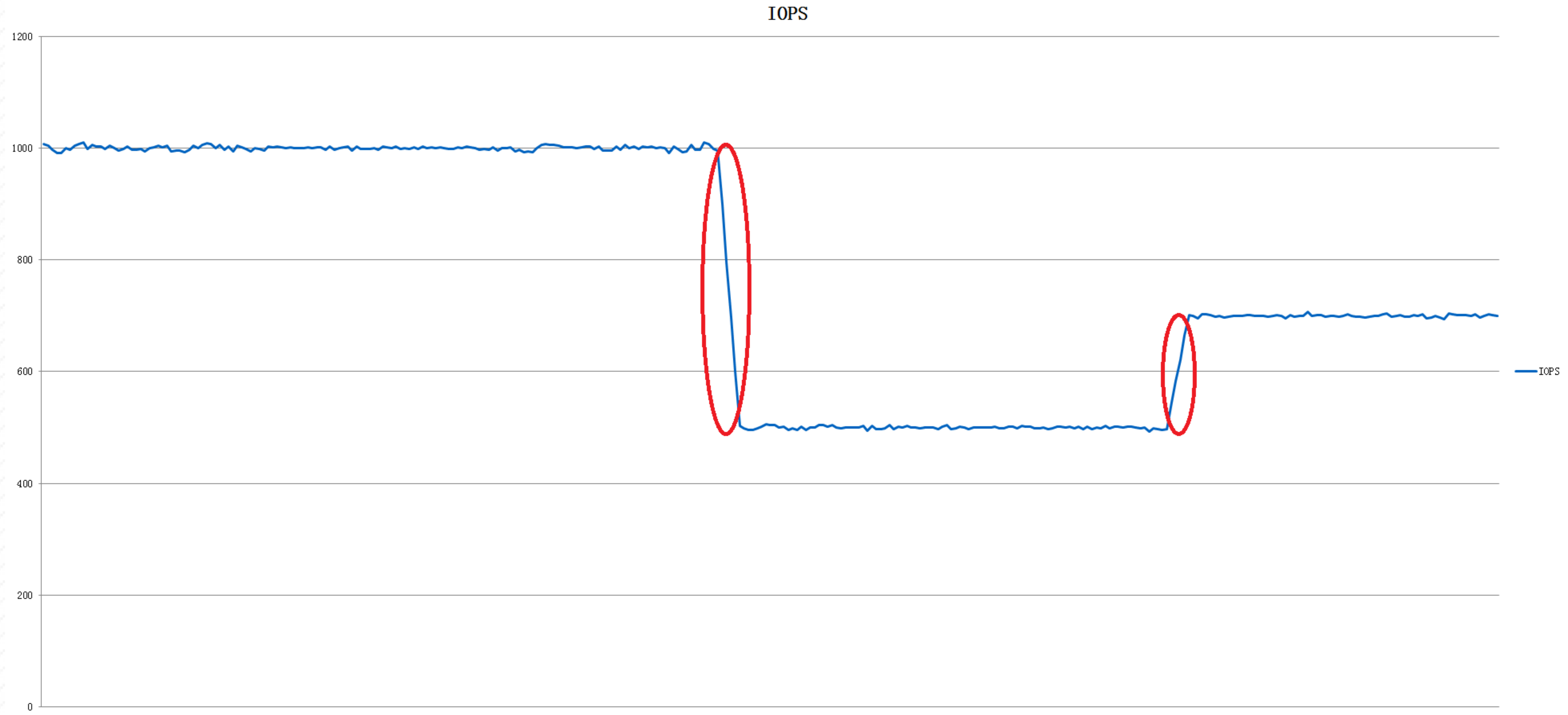
- Context *ctx
- tokens_requested

ImageRequestWQ


ImageRequestWQ
- TokenBucketThrottle iops_throttle
- _void_dequeue()
C_Throttle
- ImageRequest
- ImageRequestWQ
- complete()
_void_dequeue

```
template <typename I>
void *ImageRequestWQ<I>::_void_dequeue() {
  ...
if (item->needs_throttle() && iops_throttle->get(1, new C_Throttle(this, item))
  return nullptr;
  ...
}
```

测试结果



TODO

- Read vs Write
- Iops and bps
- bursting
- Other rate limiting

EasyStack
open cloud computing

THANK YOU