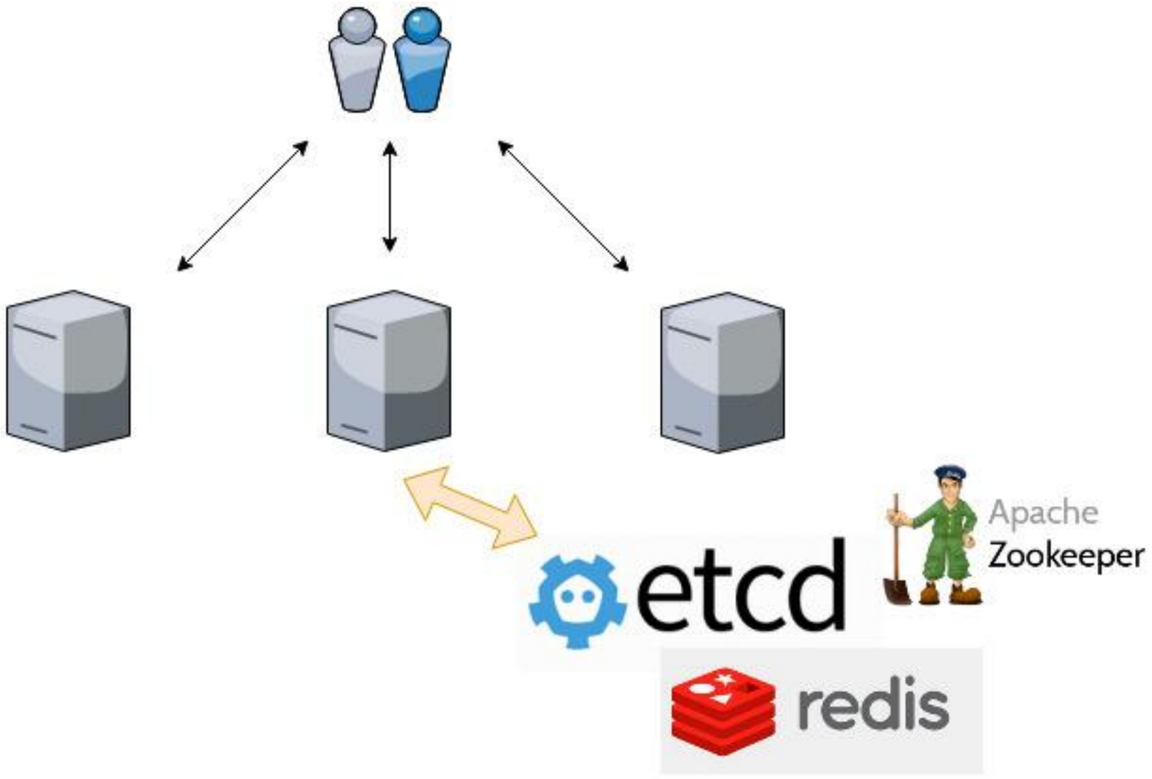
El rumor de que Python es eventualmente rapido

Token bucket, Gossip, CRDT

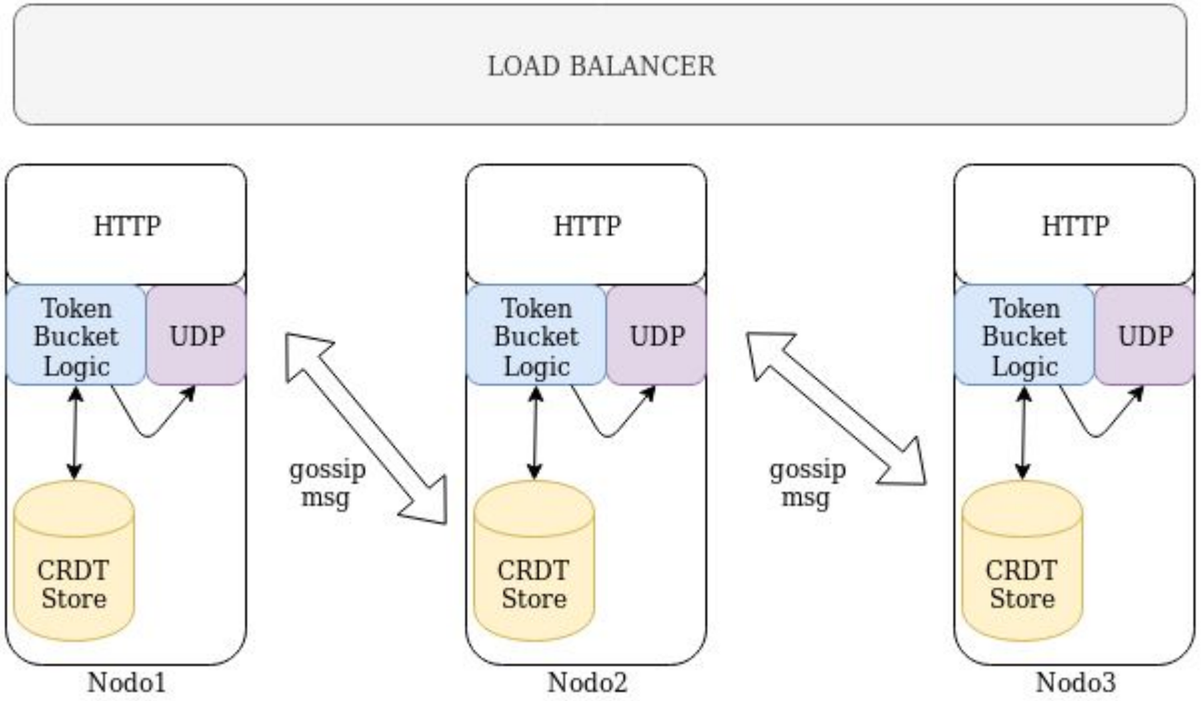
Primera idea

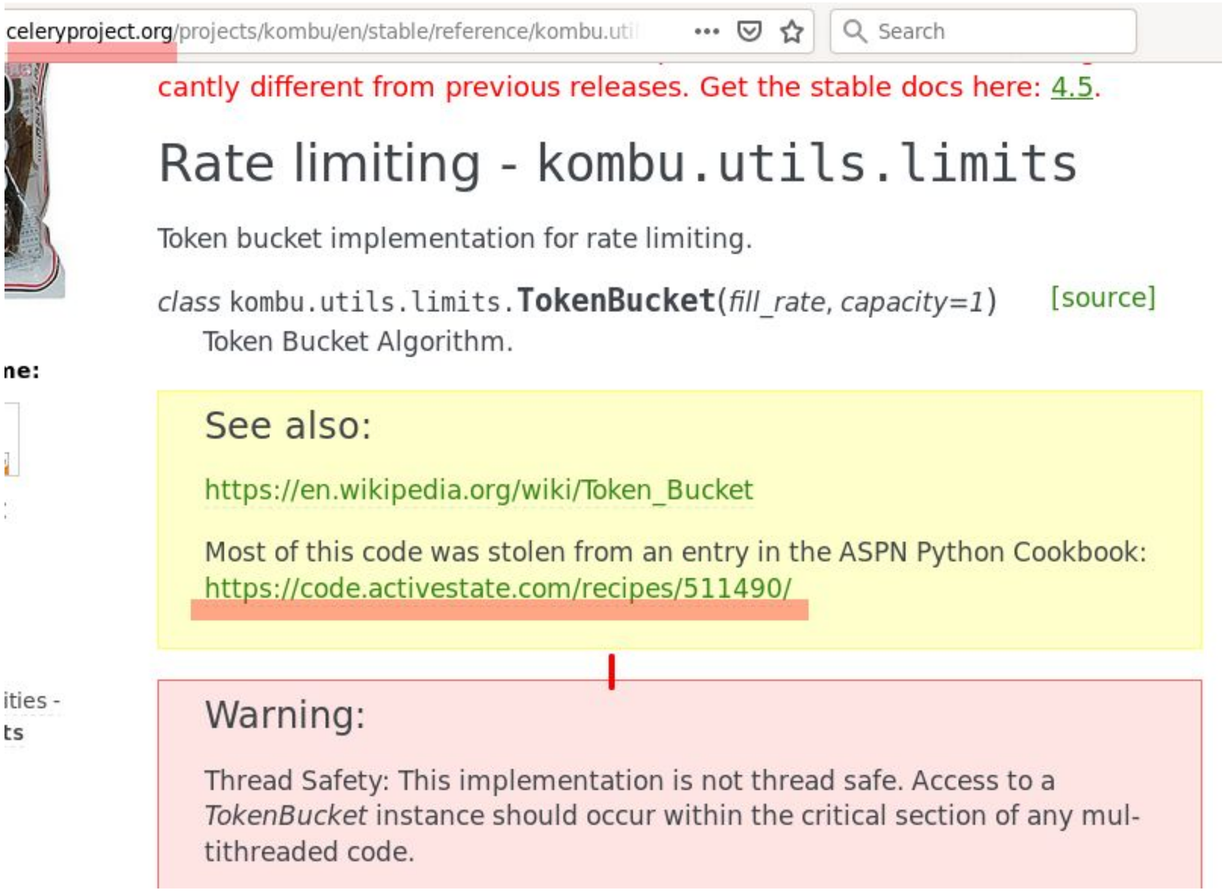


Segundo intento

Tres algoritmos

- Gossip
- CRDT (conflict-free replicated data types)
- Token Bucket





celeryproject.org/projects/kombu/en/stable/reference/kombu.uti

Search

cantly different from previous releases. Get the stable docs here: 4.5.

Rate limiting - kombu.utils.limits

Token bucket implementation for rate limiting.

class kombu.utils.limits.**TokenBucket**(*fill_rate*, *capacity=1*) [source]

Token Bucket Algorithm.

See also:

https://en.wikipedia.org/wiki/Token_Bucket

Most of this code was stolen from an entry in the ASPN Python Cookbook: https://code.activestate.com/recipes/511490/

Warning:

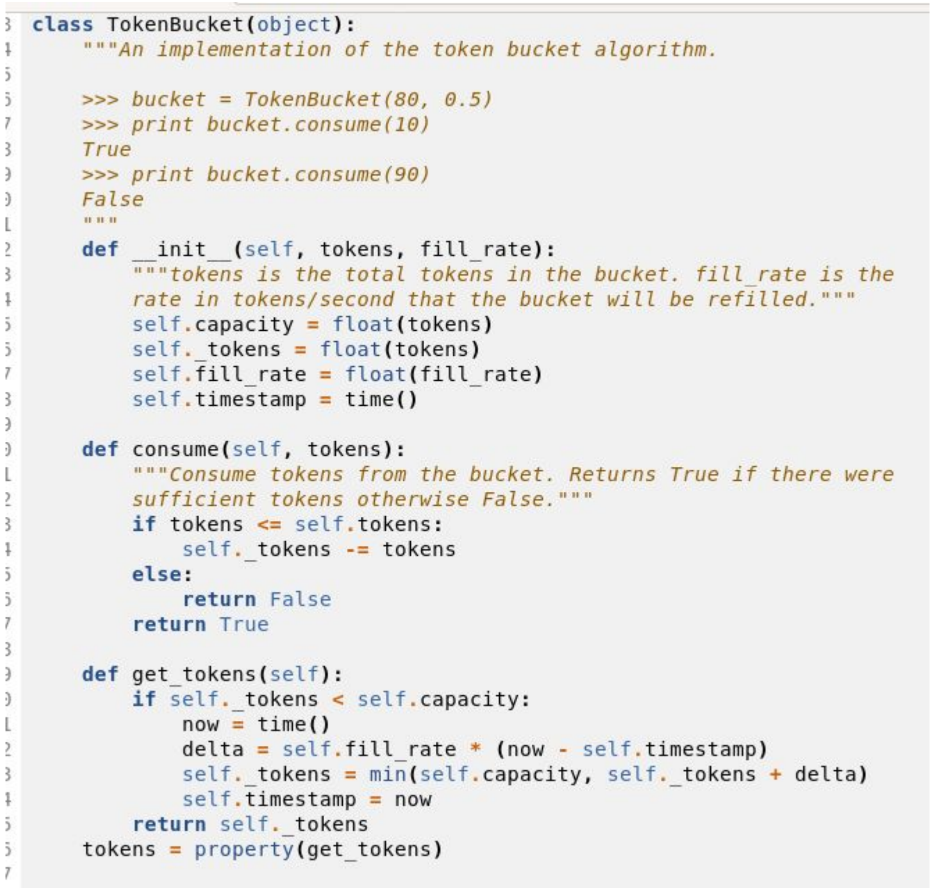
Thread Safety: This implementation is not thread safe. Access to a *TokenBucket* instance should occur within the critical section of any multithreaded code.

ne:

ities -

ts

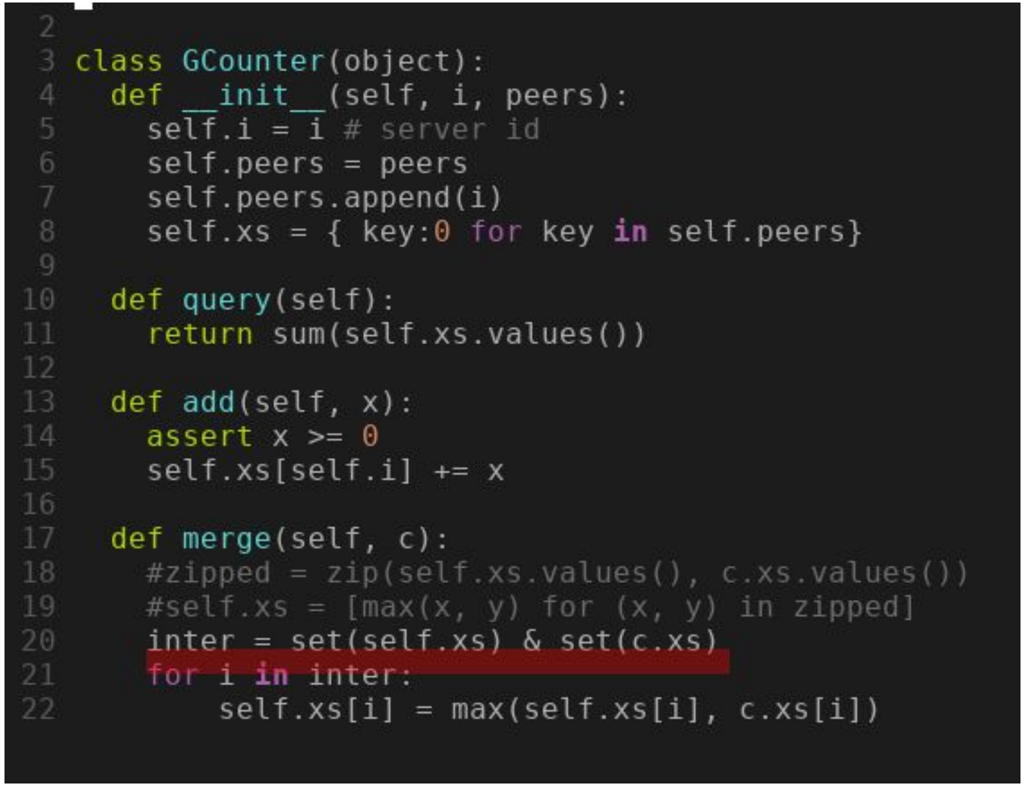
Token bucket



```python
class TokenBucket(object):
    """An implementation of the token bucket algorithm.

    >>> bucket = TokenBucket(80, 0.5)
    >>> print bucket.consume(10)
    True
    >>> print bucket.consume(90)
    False
    """
    def __init__(self, tokens, fill_rate):
        """tokens is the total tokens in the bucket. fill_rate is the
        rate in tokens/second that the bucket will be refilled."""
        self.capacity = float(tokens)
        self._tokens = float(tokens)
        self.fill_rate = float(fill_rate)
        self.timestamp = time()

    def consume(self, tokens):
        """Consume tokens from the bucket. Returns True if there were
        sufficient tokens otherwise False."""
        if tokens <= self.tokens:
            self._tokens -= tokens
        else:
            return False
        return True

    def get_tokens(self):
        if self._tokens < self.capacity:
            now = time()
            delta = self.fill_rate * (now - self.timestamp)
            self._tokens = min(self.capacity, self._tokens + delta)
            self.timestamp = now
        return self._tokens
    tokens = property(get_tokens)
```

Token bucket implementacion en python 36 lineas de codigo

CRDT / GCounter



```python

class GCounter(object):
  def __init__(self, i, peers):
    self.i = i # server id
    self.peers = peers
    self.peers.append(i)
    self.xs = { key:0 for key in self.peers}

  def query(self):
    return sum(self.xs.values())

  def add(self, x):
    assert x >= 0
    self.xs[self.i] += x

  def merge(self, c):
    #zipped = zip(self.xs.values(), c.xs.values())
    #self.xs = [max(x, y) for (x, y) in zipped]
    inter = set(self.xs) & set(c.xs)
    for i in inter:
      self.xs[i] = max(self.xs[i], c.xs[i])
```

Demo

Gracias!

Links y contacto:

- Repo
  https://github.com/nuxion/simplepy
- http://archagon.net/blog/2018/03/24/data-laced-with-history/
- https://mwhittaker.github.io/consistency_in_distributed_systems/3_crdt.html
- https://code.activestate.com/recipes/511490/

Telegram: @xpetit / Twitter: @xpetitde