滑动窗口

数据载入

对于本项目来说，数据就是日志的一行行记录，载入数据就是文件IO的读取。将获取数据的方法封装成函数。

```
def load(path):
    """装载日志文件"""
    with open(path) as f:
        for line in f:
            fields = extract(line)
            if fields:
                yield fields
            else:
                continue # TODO 解析失败就抛弃，或者打印日志
```

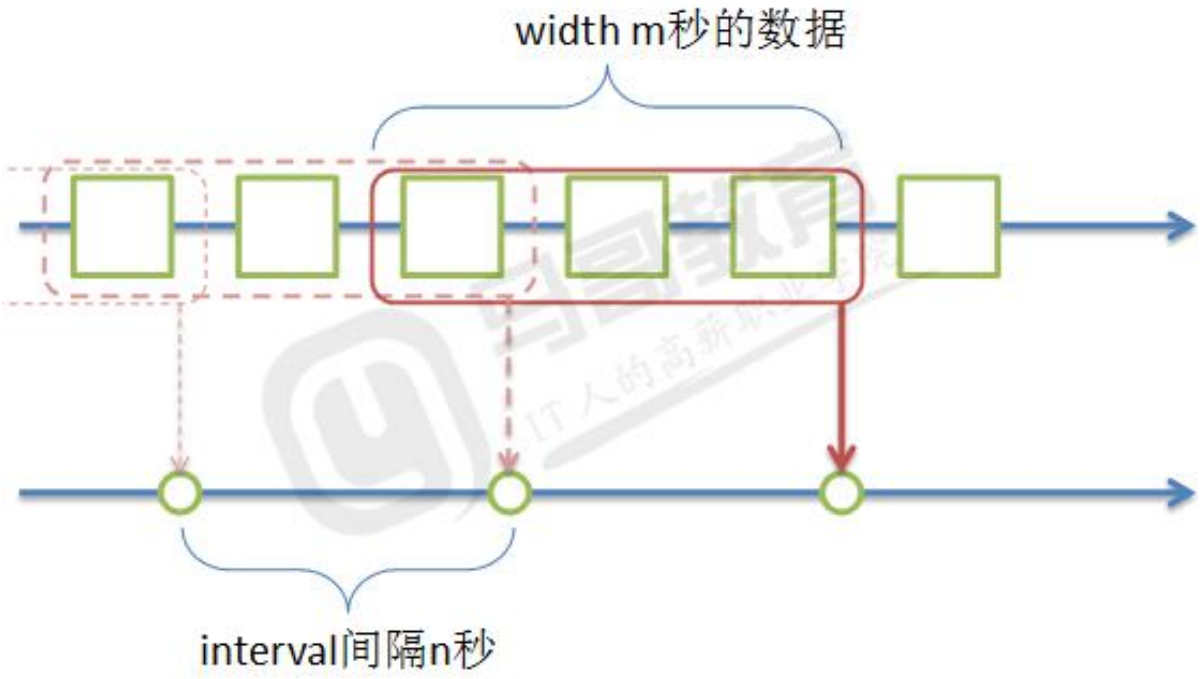

时间窗口分析

概念

很多数据，例如日志，都是和时间相关的，都是按照时间顺序产生的。
产生的数据分析的时候，要按照时间求值

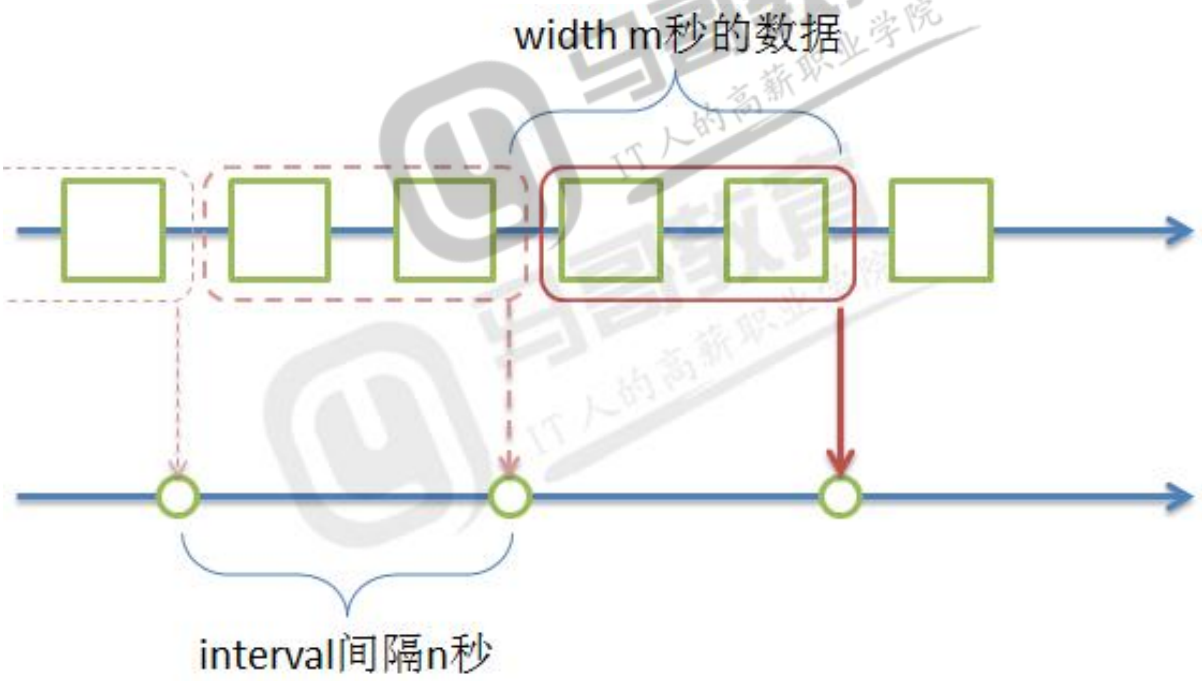
interval 表示每一次求值的时间间隔
width 时间窗口宽度，指的一次求值的时间窗口宽度

当width > interval



数据求值时会有重叠

当width = interval



数据求值没有重叠

当width < interval

一般不采纳这种方案，会有数据缺失

时序数据

运维环境中，日志、监控等产生的数据都是与时间相关的数据，按照时间先后产生并记录下来的数据，所以一般按照时间对数据进行分析。

数据分析基本程序结构

无限的生成随机数函数，产生时间相关的数据，返回 时间 和 随机数的字典
每次取3个数据，求平均值。

```
import random
import datetime
import time

def source():
    while True:
        yield {'value':random.randint(1,100),'datetime':datetime.datetime.now()}
        time.sleep(1)

# 获取数据
s = source()
items = [next(s) for _ in range(3)]

# 处理函数
def handler(iterable):
    return sum(map(lambda item:item['value'], iterable)) / len(iterable)

print(items)
print("{:.2f}".format(handler(items)))
```

上面代码模拟了，一段时间内产生了数据，等了一段固定的时间取数据来计算平均值。

窗口函数实现

将上面的获取数据的程序扩展为window函数。使用重叠的方案。

```
import random
import datetime
import time
```

```
def source(second=1):
    """生成数据"""
    while True:
        yield {
            'datetime':datetime.datetime.now(datetime.timezone(datetime.timedelta(h
ours=8))),
            'value':random.randint(1,100)
        }
        time.sleep(second)


def window(iterator, handler, width:int, interval:int):
    """
    窗口函数

    :param iterator: 数据源，生成器，用来拿数据
    :param handler: 数据处理函数
    :param width: 时间窗口宽度，秒
    :param interval: 处理时间间隔，秒
    """

    start = datetime.datetime.strptime('20170101 000000 +0800', '%Y%m%d %H%M%S %z')
    current = datetime.datetime.strptime('20170101 010000 +0800', '%Y%m%d %H%M%S %z
')
    buffer = [] # 窗口中的待计算数据
    delta = datetime.timedelta(seconds=width-interval)

    while True:
        # 从数据源获取数据
        data = next(iterator)
        if data:
            buffer.append(data) # 存入临时缓冲等待计算
            current = data['datetime']

        # 每隔interval计算buffer中的数据一次
        if (current - start).total_seconds() >= interval:
            ret = handler(buffer)
            print('{:.2f}'.format(ret))
            start = current
```

```
            # 清除超出width的数据
            buffer = [x for x in buffer if x['datetime'] > current - delta]

def handler(iterable):
    return sum(map(lambda x:x['value'], iterable)) / len(iterable)

window(source(), handler, 10, 5)
```

时间的计算