Sending million pushes

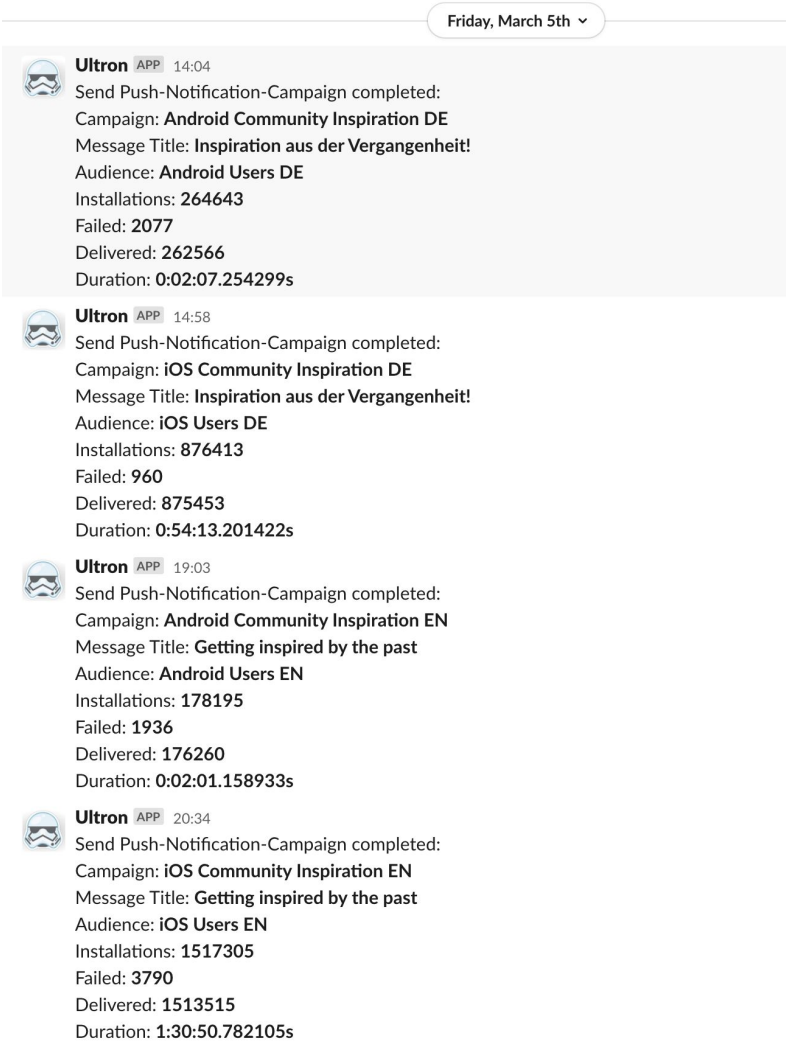
with threadpoolexecutor

Requirements

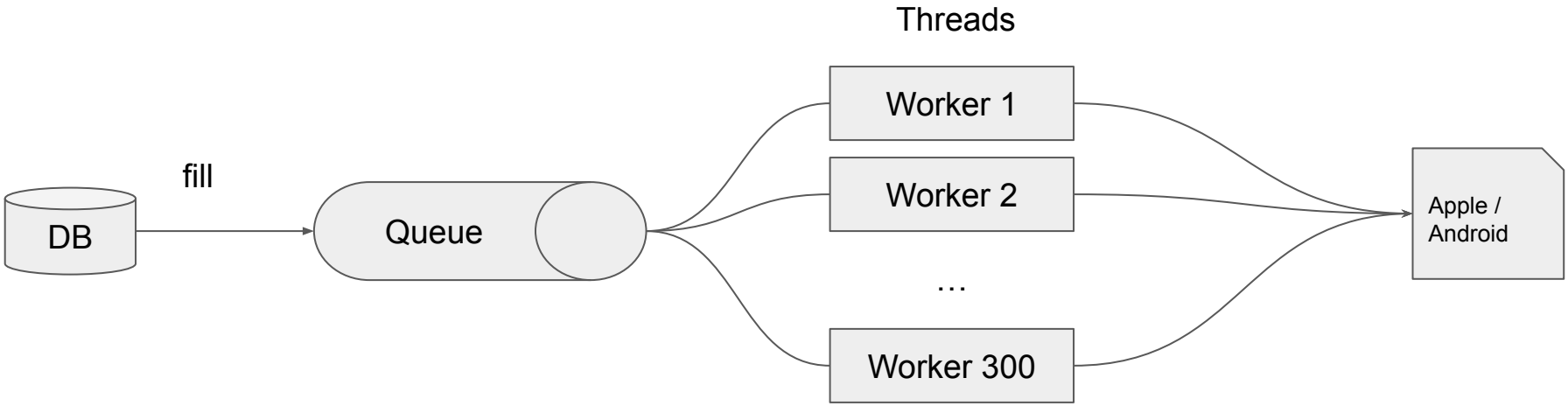
- send batch pushes to android / iOs.
- A campaign sends to one platform and language.
- send them as fast as possible.
- Track when a send failed and persist it.

Pitfalls

- Apple allows max 300-http2 connections
- getting millions of row from PostgreSQL need another approach of fetching data.
- django-push-notification couldn’t handle ~1.5mio pushes that time.
- APNS no send_batch



Initial implementation



Initial implementation

```python
class SenderThread(threading.Thread):
    def __init__(self, thread_id, service):
        super(SenderThread, self).__init__()

        self.thread_id = thread_id
        self.queue = service.queue
        self.service = service
        self.client = service.get_client()

    def run(self):
        while not self.service.thread_exit_flag:
            self.service.queue_lock.acquire()
            if not self.queue.empty():
                insta_id, push_token = self.queue.get()
                self.service.queue_lock.release()
                self.service.send_push_notification_to(
                    installation_id,
                    push_token
                )
            else:
                self.service.queue_lock.release()
```

```python
class Service:
    # stuf

    def fill_queue(self):
        installations_qs = self.get_queryset()
        self.installation_count = installations_qs.count()

        self.queue = queue.Queue(self.installation_count)

        # Fill Queue
        self.queue_lock.acquire()
        for installation in installations_qs.iterator():
            self.queue.put(installation)
        self.queue_lock.release()


    def create_worker_threads(self):
        for e in range(300):
            fret = SenderThread(e, self)
            self.threads.add(fret)
            fret.start()

        # Wait for queue to emtpy
        while not self.queue.empty():
            pass

        # Queue is empty threads can exit now
        self.thread_exit_flag = True

        # Wait for all threads to complete
        for fret in self.threads:
            fret.join()
```

Threadpool implementation

- Make use of `threading.local()` for thread-specific variable.
- Attach object to `threading_local`
- Queue management and locking is managed by `ThreadPoolExecutor`.
- Just handover the function and params.
- function can also be a class function with context of a class.

```python
import threading


thread_local = threading.local()


def send_push(installation, payload):
    if not hasattr(thread_local, 'client'):
        thread_local.client = APNSClient(**conf)

    thread_local.client.send_notification(
        push_token=installation.token, payload=payload
    )


with ThreadPoolExecutor(max_workers=300) as ex:
    installations_qs, self.installation_count = self.get_queryset()

    for installation in installations_qs.iterator():
        ex.submit(send_push, installation, payload)

    ex.shutdown(wait=True)

```

Threadpool in python

```
155     def submit(self, fn, /, *args, **kwargs):
156         with self._shutdown_lock, _global_shutdown_lock:
157             if self._broken:
158                 raise BrokenThreadPool(self._broken)
159
160             if self._shutdown:
161                 raise RuntimeError('cannot schedule new futures after shutdown')
162             if _shutdown:
163                 raise RuntimeError('cannot schedule new futures after '
164                                    'interpreter shutdown')
165
166             f = _base.Future()
167             w = _WorkItem(f, fn, args, kwargs)
168
169             self._work_queue.put(w)
170             self._adjust_thread_count()
171             return f
172     submit.__doc__ = _base.Executor.submit.__doc__
173
```

```
210     def shutdown(self, wait=True, *, cancel_futures=False):
211         with self._shutdown_lock:
212             self._shutdown = True
213             if cancel_futures:
214                 # Drain all work items from the queue, and then cancel their
215                 # associated futures.
216                 while True:
217                     try:
218                         work_item = self._work_queue.get_nowait()
219                     except queue.Empty:
220                         break
221                     if work_item is not None:
222                         work_item.future.cancel()
223
224             # Send a wake-up to prevent threads calling
225             # _work_queue.get(block=True) from permanently blocking.
226             self._work_queue.put(None)
227         if wait:
228             for t in self._threads:
229                 t.join()
230     shutdown.__doc__ = _base.Executor.shutdown.__doc__
```

Tank you

contact:
Linkedin:
https://www.linkedin.com/in/trung-phan-869a47130/