

Redis como *Middleware* Orientado a Mensajes (MoM)

Maykel Moya

Máster en Sistemas Telemáticos e Informáticos
Universidad Rey Juan Carlos
Madrid

abril 2012

©2012 Maykel Moya
Algunos derechos reservados
Este trabajo se distribuye bajo la licencia



Creative Commons Attribution-NonCommercial-ShareAlike
disponible en <http://creativecommons.org/licenses/by-nc-sa/3.0/deed.es>

Agenda

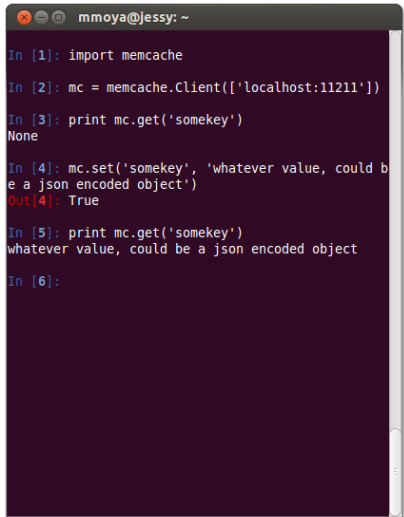
- 1 Memcached
- 2 Redis
- 3 Redis como MoM
- 4 Demo

En el principio fue Memcached

- Servidor de caché (acceso por la red)
- Sólo comandos GET y SET (simple)
- Alto rendimiento
- Almacenamiento en memoria (volátil)
- Almacenamiento basado en llave/valor
- Software Libre
- <http://memcached.org/>

Uso típico

Sólo comandos GET y SET

A terminal window titled 'mmoya@jessy: ~' with a dark purple background. It displays a series of Python commands and their outputs for interacting with a memcache client. The commands are numbered 1 through 6. The output for command 3 is 'None', for command 4 is 'True', and for command 5 is the string 'whatever value, could be a json encoded object'.

```
In [1]: import memcache
In [2]: mc = memcache.Client(['localhost:11211'])
In [3]: print mc.get('somekey')
None
In [4]: mc.set('somekey', 'whatever value, could be a json encoded object')
Out[4]: True
In [5]: print mc.get('somekey')
whatever value, could be a json encoded object
In [6]:
```

Uso típico

Sólo comandos GET y SET

```
mmoya@jessy: ~  
  
In [1]: import memcache  
  
In [2]: mc = memcache.Client(['localhost:11211'])  
  
In [3]: print mc.get('somekey')  
None  
  
In [4]: mc.set('somekey', 'whatever value, could be  
a json encoded object')  
Out[4]: True  
  
In [5]: print mc.get('somekey')  
whatever value, could be a json encoded object  
  
In [6]:
```

```
1  #!/usr/bin/python  
2  # -*- coding: utf-8 -*-  
3  
4  from __future__ import print_function  
5  
6  import memcache  
7  import time  
8  
9  mc = memcache.Client(['localhost:11211'])  
10  
11  def expensive_hello(s):  
12      time.sleep(2)  
13      return 'Hello %s!' % s  
14  
15  def cache_hello():  
16      key = 'hello'  
17  
18      value = mc.get(key)  
19      if value is None:  
20          value = expensive_hello('world')  
21          mc.set(key, value)  
22  
23      return value  
24  
25  value = cache_hello()  
26  print(value)
```

Luego Redis

- Servidor de caché (acceso por la red)
- Alto rendimiento
- Almacenamiento persistente
- Almacenamiento basado en llave/valor
 - Pero los valores pueden ser *strings*, *hashes*, *lists* entre otros
 - ... por lo que se conoce también como Servidor de Estructuras de Datos
- Software Libre
- <http://redis.io/>

Comandos

- Además de los básicos GET y SET
- LPUSH: Inserción de un elemento al principio de una lista (*Left PUSH*)
- RPOP: Eliminación de un elemento del final de una lista (*Right POP*)
- BRPOP: Idem pero bloqueante (*Blocking Right POP*)
- PUBLISH: Publicación en un canal
- SUBSCRIBE: Suscripción a un canal
- UNSUBSCRIBE: Cancelar suscripción a un canal
- *muchos otros*
- <http://redis.io/commands>

Middleware Orientado a Mensajes (MoM)

De Wikipedia:

... infraestructura de software o hardware que permite el envío y recepción de mensajes entre componentes de un sistema distribuido

Paradigmas de distribución en un MoM

	cuántos suscrip- tores reciben el mensaje?	se retiene el mensaje?	se puede imple- mentar con Re- dis?
<i>queue</i>	uno	sí	sí
<i>topic</i>	todos	no	sí
<i>durable subscription</i>	todos	sí	sí

Paradigmas de distribución en un MoM

	cuántos suscrip- tores reciben el mensaje?	se retiene el mensaje?	se puede imple- mentar con Re- dis?
<i>queue</i>	uno	sí	sí
<i>topic</i>	todos	no	sí
<i>durable subscription</i>	todos	sí	sí

¿Cómo se implementa con Redis?

- *Queue*: Con LPUSH y BRPOP

Paradigmas de distribución en un MoM

	cuántos suscrip- tores reciben el mensaje?	se retiene el mensaje?	se puede imple- mentar con Re- dis?
<i>queue</i>	uno	sí	sí
<i>topic</i>	todos	no	sí
<i>durable subscription</i>	todos	sí	sí

¿Cómo se implementa con Redis?

- *Queue*: Con LPUSH y BRPOP
- *Topic*: Con SUBSCRIBE y PUBLISH

Paradigmas de distribución en un MoM

	cuántos suscrip- tores reciben el mensaje?	se retiene el mensaje?	se puede imple- mentar con Re- dis?
<i>queue</i>	uno	sí	sí
<i>topic</i>	todos	no	sí
<i>durable subscription</i>	todos	sí	sí

¿Cómo se implementa con Redis?

- *Queue*: Con LPUSH y BRPOP
- *Topic*: Con SUBSCRIBE y PUBLISH
- *Durable Subscription*: Con SUBSCRIBE, PUBLISH y *Sorted Sets*, ver <http://j.mp/JBjwdT>.

Y ahora el demo!



- ❶ Choosing a message queue for Python on Ubuntu on a VPS.
- ❷ Is non-blocking Redis pubsub possible?.
- ❸ Redis Pub/Sub.
- ❹ Does the redis pub/sub model require persistent connections to redis?.
- ❺ Redis as Messaging Middleware.
- ❻ Do you need a Push Notification Manager? – Redis PubSub to the rescue.