actibyti] a netmind social project

Redis

© 2017, ACTIBYTI PROJECT SLU, Barcelona
Autor: Ricardo Ahumada

GOBIERNO DE ESPAÑA
MINISTERIO DE ENERGÍA, TURISMO Y AGENDA DIGITAL
red.es
ESTRATEGIA DE EMPRENDIMIENTO Y EMPLEO JOVEN garantía juvenil
Agenda Digital para España
UNIÓN EUROPEA
Fondo Social Europeo
"El FSE invierte en tu futuro"

Redis

- Redis (https://redis.io/) es una de las bases de datos NoSQL más populares, y seguramente la más utilizada dentro del segmento de las bases de datos clave-valor.
- La popularidad de Redis se debe en gran medida a su espectacular velocidad, ya que mantiene la información en memoria (persiste en disco); pero también a su sencillez de uso y flexibilidad.
- Además de permitir asociar valores de tipo string a una clave, permite utilizar tipos de datos avanzados, que junto a las operaciones asociadas a estos tipos de datos, logra resolver muchos casos de uso de negocio, que a priori no pensaríamos que fuéramos capaces con una base de datos clave-valor.
- No en vano, Redis se define usualmente como un servidor de estructuras de datos, aunque en sus inicios, haciendo honor a su nombre, fuese un simple diccionario remoto (REmote DIctionary Server).

¿Para que puedo usar Redis?

- Caché de páginas web
  - Podemos usar Redis como caché de páginas HTML, o fragmentos de estas, lo que acelerará el acceso a las mismas, a la vez que evitamos llegar a los servidores web o de aplicaciones, reduciendo la carga en estos y en los sistemas de bases de datos a los que los mismos accedan.

- Almacenamiento de sesiones de usuario
  - Podemos usar Redis como un almacén de sesiones de muy rápido acceso, en el que mantengamos el identificador de sesión junto con toda la información asociada a la misma. Además de reducir los tiempos de latencia de nuestra solución, igual que en el caso anterior evitaremos una vez más accesos a otras bases de datos.

¿Para que puedo usar Redis?

- Almacenamiento de carritos de la compra
  - De forma muy similar al almacén de sesiones de usuario, podemos almacenar en Redis los artículos contenidos en la cesta de la compra de un usuario, y la información asociada a los mismos, permitiéndonos acceder en cualquier momento a ellos con una latencia mínima.

- Caché de base de datos
  - Otra forma de descargar a las bases de datos operacionales es almacenar en Redis el resultado de determinadas consultas que se ejecuten con mucha frecuencia, y cuya información no cambia a menudo, o no es crítico mantener actualizada al instante.

¿Para que puedo usar Redis?

- Contadores y estadísticas
  - Para muchos casos de uso es necesario manejar contadores y estadísticas en tiempo real, y Redis tiene soporte para la gestión concurrente y atómica de los mismos. Algunos ejemplos posibles serían el contador de visualización de un producto, votos de usuarios, o contadores de acceso a un recurso para poder limitar su uso.

- Listas de elementos recientes
  - Es muy habitual mostrar listas en las que aparecen las últimas actualizaciones de algún elemento hechas por los usuarios. Por ejemplo, los últimos comentarios sobre un post, las últimas imágenes subidas, los artículos de un catálogo vistos recientemente, etc. Este tipo de operaciones suele ser muy costoso para las bases de datos relacionales, sobre todo cuando el volumen de información se va haciendo mayor, pero Redis es capaz de resolver esta operación con independencia del volumen.

¿Para que puedo usar Redis?

- Base de datos principal
  - Para determinados casos, Redis se puede usar como almacenamiento principal gracias a la potencia de modelado que permiten sus avanzados tipos de datos. Destaca su uso en casos como los microservicios, en los que podemos aprovechar la velocidad de Redis para construir soluciones especializadas, simples de implementar y mantener, que a la vez ofrecen un alto rendimiento.

Cuando no usar Redis

- Redis es una opción terrible para…
  - Relaciones de datos complejas
  - Búsquedas

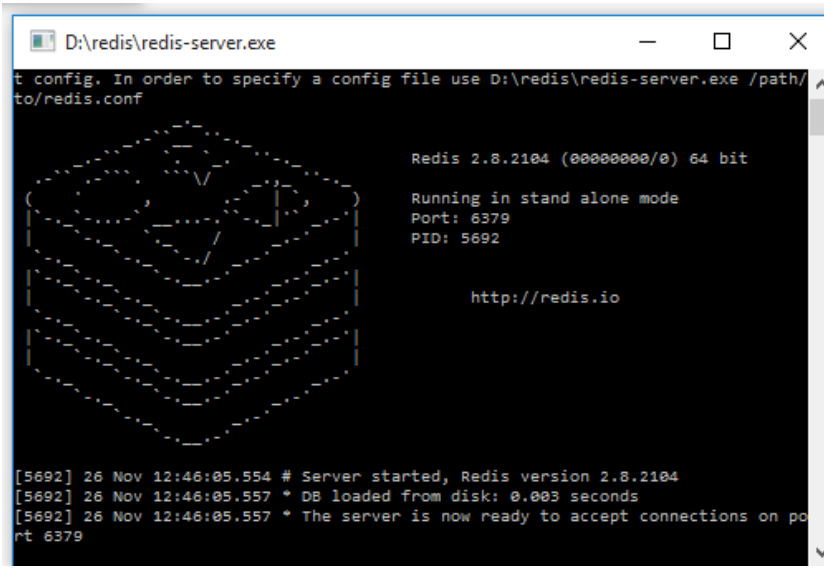
¿Quién usa Redis?

- **Twitter** usa Redis para mantener el timeline de sus usuarios. El timeline es una lista de los tuits de las personas a las que se sigue, y Twitter usa Redis para poder actualizar en el menor tiempo posible los timelines de todos sus usuarios. Teniendo en cuenta que algunos usuarios tienen más de 60.000.000 de seguidores, actualizar todas estas listas cada vez que se escribe un tuit, se convierte en una operación crítica.
- **Hulu** la usa para mantener la posición en la que un usuario se encuentra en un vídeo, así como el histórico de visualizaciones de los usuarios.
- **Pinterest** utiliza Redis para mantener la información de los usuarios y los tablones que sigue cada uno.
- **Flickr** la utiliza como base del sistema de colas para mantener de manera asíncrona las tareas a realizar sobre las imágenes de los usuarios. También forma parte de la infraestructura para la generación de notificaciones push a sus usuarios.
- **Trello** usa Redis para mantener toda la información efímera que es necesario compartir entre todos sus servidores.

Instalando Redis

Instala redis

- Descarga la última versión estable portable (zip) de https://github.com/MicrosoftArchive/redis/releases
- Descomprime el zip
- Abre un terminal de consola en el directorio y escribe
  - redis-server.exe



Accede con el cli de redis

- Abre otro terminal de consola en la carpeta descomprimida de redis
- Ejecuta
  - redis-cli.exe
- Ya puedes comenzar a usar redis

Comandos redis

- Accede al sitio de documentación redis:
  - https://redis.io/commands
- Juega con los ejemplos de SET y GET
  - https://redis.io/commands/set
  - https://redis.io/commands/get

Usando Redis con Node.js

- Node_redis, es el cliente de Redis para Node.js. Puede instalarlo en un proyecto node usando el npm:

```
npm install redis
```

- Una vez que haya instalado el modulo de node_redis ya estará listo para ser utilizado.

Usando Redis desde un proyecto Node

Conectando Redis con Node.js y configuración

- Vamos a crear un archivo simple, app.js, y ver cómo conectarse con Redis desde Node.js.

```
var redis = require('redis');
var client = redis.createClient(); //creates a new client
```

- De forma predeterminada, redis.createClient () utilizará 127.0.0.1 y 6379 como nombre de host y puerto, respectivamente.
- Si tiene un host / puerto diferente, puede suministrarlos de la siguiente manera:

```
var client = redis.createClient(port, host);
```

Conectando Redis con Node.js y configuración

- Ahora, que se ha establecido la conexión se pueden realizar las acciones. Básicamente, sólo tiene que escuchar los eventos de conexión como se muestra a continuación.

```
client.on('connect', function() {
    console.log('connected');
});
```

- El siguiente fragmento de codigo captura el evento de conexión para mostrar un mensaje en consola:

```
var redis = require('redis');
var client = redis.createClient();

client.on('connect', function() {
    console.log('connected');
});
```

- Ahora, escriba `Node app` en el terminal para ejecutar la aplicación.
- Asegúrese de que su servidor Redis esté funcionando antes de ejecutar el codigo.

Almacenamiento de pares clave-valor

- Ahora que sabe cómo conectarse con Redis desde Node.js, vamos a almacenar pares clave-valor en el almacenamiento Redis.

- Almacenamiento de cadenas
  - Todos los comandos Redis se exponen como diferentes funciones en el objeto cliente. Para almacenar una cadena simple, utilice la siguiente sintaxis:

```
client.set('framework', 'AngularJS');
```

O

```
client.set(['framework', 'AngularJS']);
```

Almacenamiento de pares clave-valor

- Los fragmentos anteriores almacenan una cadena simple “AngularJS” en la clave “framework”.
- Se debe tener en cuenta que ambos fragmentos hacen lo mismo. La única diferencia es que el primero pasa un número variable de argumentos mientras que el último pasa una matriz args a la función client.set ().
- También puede pasar una función de callback procesar la respuesta cuando se complete la operación:

```
client.set('framework', 'AngularJS', function(err, reply) {
  console.log(reply);
});
```

- Si la operación falló por alguna razón, el argumento **err** representa el error.

Almacenamiento de pares clave-valor

- Para recuperar el valor de una clave, usamos el método **get**:

```
client.get('framework', function(err, reply) {
    console.log(reply);
});
```

- **Client.get ()** permite recuperar una clave almacenada en Redis.
- El valor de la clave se puede acceder a través de la respuesta del argumento de la devolución de llamada.
- Si la clave no existe, el valor de la respuesta estará vacío.

Almacenamiento de hash

- Muchas veces el almacenamiento de valores simples no resolverá su problema.
- Tendrá que almacenar hashes (objetos) en Redis. Para ello puede utilizar la función hmset() como sigue:

```
client.hmset('frameworks', 'javascript', 'AngularJS', 'css', 'Bootstrap', 'node', 'Express');

client.hgetall('frameworks', function(err, object) {
    console.log(object);
});
```

- El fragmento anterior almacena un hash en Redis que asigna cada tecnología a su framework.
  - El primer argumento para hmset () es el nombre de la clave. Los argumentos posteriores representan pares clave-valor.
  - Similarmente, hgetall () se utiliza para recuperar el valor de la clave. Si se encuentra la clave, el segundo argumento de la devolución de llamada contendrá el valor (que es un objeto).

Almacenamiento de pares clave-valor

- También puede utilizar la siguiente sintaxis para almacenar objetos en Redis:

```
client.hmset('frameworks', {
    'javascript': 'AngularJS',
    'css': 'Bootstrap',
    'node': 'Express'
});
```

- Similarmente se puede añadir una función de callback para saber cuándo se ha completado la operación.

Almacenamiento de listas

- Si desea almacenar una lista de elementos, puede utilizar listas Redis. Para almacenar una lista se utiliza **rpush**:

```
client.rpush(['frameworks', 'angularjs', 'backbone'], function(err, reply) {
    console.log(reply); //prints 2
});
```

- El fragmento anterior crea una lista llamada “marcos” y añade dos elementos a la misma. Por lo tanto, la longitud de la lista es ahora dos.
- Se ha pasado una matriz args a rpush.
  - El primer elemento de la matriz representa el nombre de la clave mientras que el resto representa los elementos de la lista.
  - También puede utilizar lpush () en lugar de rpush () para añadir los elementos hacia la izquierda.

Almacenamiento de listas

- Para recuperar los elementos de la lista puede utilizar la función **lrange**() como sigue:

```
client.lrange('frameworks', 0, -1, function(err, reply) {
    console.log(reply); // ['angularjs', 'backbone']
});
```

- Se obtienen todos los elementos de la lista pasando -1 como el tercer argumento a **lrange**().
- Si se desea un subconjunto de la lista, debe pasar el índice final en su lugar.

Almacenamiento de Conjuntos

- Los conjuntos son similares a las listas, pero la diferencia es que no permiten valores duplicados. Por lo tanto, si no se desea ningún elemento duplicado en su lista puede utilizar un conjunto.
- Podemos modificar nuestro fragmento anterior para usar un conjunto en lugar de una lista.

```
client.sadd(['tags', 'angularjs', 'backbonejs', 'emberjs'], function(err, reply)
{
    console.log(reply); // 3
});
```

Almacenamiento de Conjuntos

- Para recuperar los miembros del conjunto, utilice la función **smembers**() como sigue:

```
client.smembers('tags', function(err, reply) {
    console.log(reply);
});
```

- Este fragmento recuperará todos los miembros del conjunto.
- Se debe tener en cuenta que el orden no se conserva mientras recupera los miembros.

Comprobación de la existencia de claves

- Para comprobar si una clave ya existe se puede utilizar la función **exists**() como se muestra a continuación:

```
client.exists('key', function(err, reply) {
    if (reply === 1) {
        console.log('exists');
    } else {
        console.log('doesn\'t exist');
    }
});
```

Eliminación y caducidad de claves

- En muchas ocaciones se necesitará borrar algunas claves y reinicializarlas.
- Para borrar las claves, se puede usar el comando **del**:

```
client.del('frameworks', function(err, reply) {
    console.log(reply);
});
```

- También puede signarun tiempo de caducidad a una clave existente:

```
client.set('key1', 'val1');
client.expire('key1', 30);
```

  - El fragmento anterior asigna un tiempo de expiración de 30 segundos a la clave key1.

Incrementando y decrementando

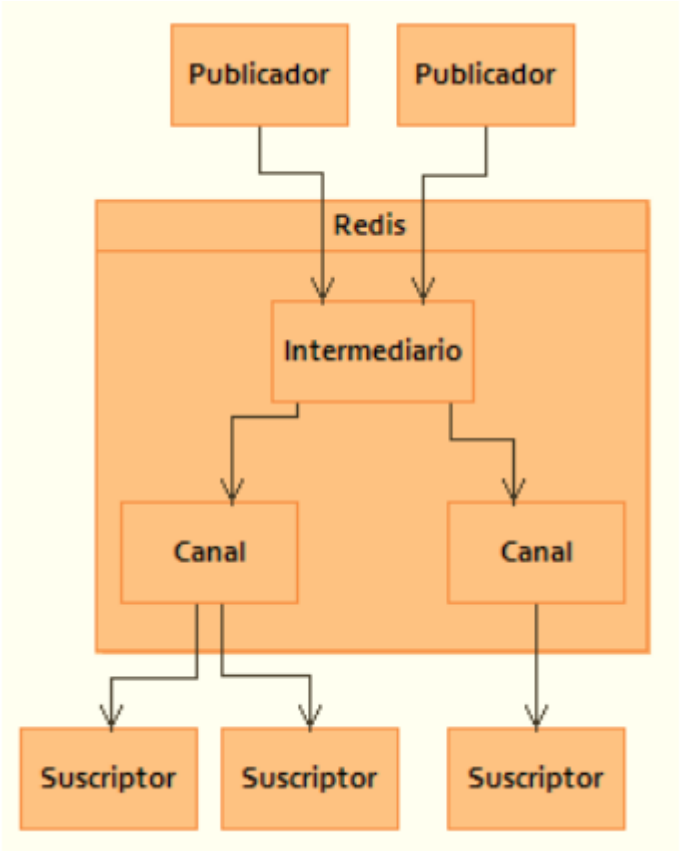
- Redis también admite de incremento y decremento de valores de una clave.
- Para incrementar la función **incr()** de una función de clave, como se muestra a continuación:

```
client.set('key1', 10, function() {
    client.incr('key1', function(err, reply) {
        console.log(reply); // 11
    });
});
```

- La función incr() incrementa un valor de clave en 1. Si necesita incrementar en una cantidad diferente, puede utilizar la función incrby().
- Del mismo modo, para decrementar una clave puede utilizar las funciones como decr() y decrby().

Redis – Mensajería

- Una característica muy valorada de Redis es su servicio de mensajería integrado que integra el modelo publicación/suscripción (publish/subscribe).
- Un canal (channel) o tema (topic) es un contenedor donde se publican mensajes. Es el medio a través del cual los extremos intercambian mensajes.
- El intermediario es el responsable de administrarlos. Una vez los ha entregado, los suprime automáticamente.
- Los emisores de los mensajes se los envían al intermediario indicándole a través de qué canales debe difundirlos (publication).
- Mientras que cuando un componente desea recibir mensajes publicados por otros lo que hace es abonarse a los canales que son de su interés (subscription).
- Cada vez que el intermediario publica un mensaje en un canal, cuando pueda se lo enviará a sus suscriptores y, finalizado, lo suprimirá del canal.



Redis – Mensajería con CLI

- Usando el cli de redis podemos publicar y suscribirnos a canales.
- Publicar
  - redis-cli
  - > PUBLISH channel message
  - > PUBLISH alta "{'correo': 'elvis@costello.com', 'nombre': 'Elvis'}"
- Suscribirse/De-suscribirse (se puede suscribir a más de uno)
  - redis-cli
  - > SUBSCRIBE canal canal canal…
  - > UNSUBSCRIBE canal canal canal…
  - > SUBSCRIBE alta

Redis – PUB/SUB

- El comando PUBSUB se utiliza para obtener información del servicio de mensajería: los canales existentes y los suscriptores.
- Para conocer los canales, se utiliza la siguiente sintaxis:
  - PUBSUB CHANNELS
  - PUBSUB CHANNELS [argument]

  - Si no se indica ningún argumento, mostrará todos los existentes actualmente.
  - Si se indica un patrón, aquellos cuyo nombre coincida con el indicado.
- Para conocer el número de suscriptores de uno o más canales indicados, se utiliza:
  - PUBSUB NUMSUB canal canal canal…

  - redis-cli
  - > PUBSUB CHANNELS
  - > PUBSUB NUMSUB alta baja

Integración con Node

- Redis provee dos métodos para suscribirse y publicar en canales de mensajería (https://www.npmjs.com/package/redis)
  - publish(channel, cb)
  - subscribe(channel)
    - Eventos:
      - message
      - subscribe
      - unsubscribe
  - unsubscribe()
- Hay que tener en cuenta que un cliente cuando publica ya no puede tener otro rol, por tanto será necesario crear dos clientes para una comunicación completa

Ejemplo

```
var redis = require("redis");
var sub = redis.createClient(), pub = redis.createClient();
var msg_count = 0;

sub.on("subscribe", function (channel, count) {
    pub.publish("a nice channel", "I am sending a message.");
    pub.publish("a nice channel", "I am sending a second message.");
    pub.publish("a nice channel", "I am sending my last message.");
});

sub.on("message", function (channel, message) {
    console.log("sub channel " + channel + ": " + message);
    msg_count += 1;
    if (msg_count === 3) {
        sub.unsubscribe();
        sub.quit();
        pub.quit();
    }
});

sub.subscribe("a nice channel");
```

Persistencia de BananaTube con Redis

- Analiza en equipo si se podría usar Redis como base de datos de persistencia para BananaTube
- Que ventajas/desventajas presenta?
- Cubre todos los casos de uso posible?

netmind

WeKnowIT

Barcelona

C. Almogàvers, 123
08018 Barcelona
Tel. 93 304.17.20
Fax. 93 304.17.22

Madrid

Plaza Carlos Trías Bertrán, 7
28020 Madrid
Tel. 91 442.77.03
Fax. 91 442.77.07

www.netmind.es

GOBIERNO DE ESPAÑA
MINISTERIO DE ENERGÍA, TURISMO Y AGENDA DIGITAL
red.es
ESTRATEGIA DE EMPRENDIMIENTO Y EMPLEO JOVEN garantía juvenil
Agenda Digital para España
UNIÓN EUROPEA
Fondo Social Europeo
"El FSE invierte en tu futuro"