# Memcached

Indice

[Memcached 1](#_Toc5086322)

[¿Qué es? 1](#_Toc5086323)

[¿Quiénes usan memcached? 2](#_Toc5086324)

[¿Cómo funciona Memcached? 2](#_Toc5086325)

[Arquitectura de Memcached 3](#_Toc5086326)

[Protocolo 4](#_Toc5086327)

[Seguridad y ataques 4](#_Toc5086328)

[Ventajas 5](#_Toc5086329)

[Inconvenientes 6](#_Toc5086330)

[Datos curiosos 6](#_Toc5086331)

[Instalación 7](#_Toc5086332)



## ¿Qué es?

Memcached es un sistema de almacenamiento en caché de memoria distribuida de código abierto. Se utiliza para **acelerar las aplicaciones web dinámicas** al reducir la carga de la base de datos. En otras palabras, cada vez que se realiza una solicitud de base de datos agrega una carga adicional al servidor. Memcached reduce esa carga al almacenar objetos de datos en la memoria dinámica (memoria a corto plazo para aplicaciones). Memcached almacena datos basados ​​en valores-clave para pequeñas cadenas u objetos arbitrarios, que incluyen:

* Resultados de llamadas a bases de datos
* Llamadas API
* Representación de página

Memcached se compone de cuatro componentes principales. Estos componentes permiten que el cliente y el servidor trabajen juntos para entregar datos en caché de la manera más eficiente posible:

1. **Software de cliente** : al que se le proporciona una lista de servidores Memcached disponibles
2. **Un algoritmo de hashing basado en el cliente** : elige un servidor basado en la "clave"
3. **Software del servidor** : almacena los valores y sus claves en una tabla hash interna
4. **LRU** : determina cuándo tirar datos antiguos o reutilizar la memoria

## ¿Quiénes usan memcached?

El desarrollo inicial y sus posteriores mejoras nacen como una necesidad de incrementar velocidades de respuesta para las peticiones web, en sitios de tráfico masivo.

Actualmente, su uso continúa expandiéndose, a medida que el proyecto toma mayor fuerza con la ayuda de varios desarrolladores de la comunidad [Open Source](http://www.maestrosdelweb.com/editorial/apertura-o-libertad-resena-de-una-involucion/) que revisan y agregan nuevas capacidades al proyecto.

Memcached nace como un desarrollo para el backend de **LiveJournal**, y como tal, es el primer sitio que lo implementa. Luego de tener ciertas falencias de performance (producto de tener que manejar en promedio, de 35 a 40 millones de page views diarios) implementaron Memcached, logrando una gran mejoría en la prestación del servicio. Fue el caso de **Slashdot, Wikipedia y Fotolog.**

Algunos sitios web importantes como **Youtube, Facebook o Twitter** utilizan Memcached para optimizar el rendimiento y conseguir cachear los datos que tienen que enviar a los usuarios y visitantes.

En **Google App Engine** o en **Amazon AWS** podemos encontrar “Instancias” dedicadas a Memcached con una mínima latencia en las conexiones si utilizas como servidor de aplicación una de las “instancias” del servicio.  **[Heroku](https://es.wikipedia.org/wiki/Heroku" \o "Heroku)** ofrece un servicio de Memcached gestionado con NorthScale[8](https://es.wikipedia.org/wiki/Memcached#cite_note-8)​ como parte de su [PaaS](https://es.wikipedia.org/wiki/PaaS" \o "PaaS).

Amazon AWS: <https://aws.amazon.com/es/memcached/>

## ¿Cómo funciona Memcached?

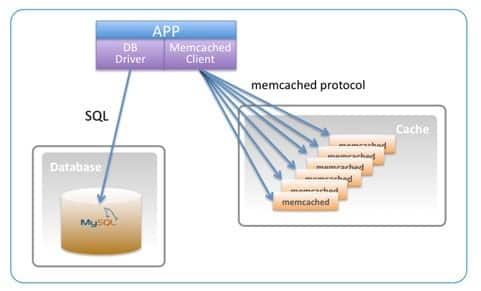
Memcached consta de cuatro componentes principales y estos componentes son los que le permiten almacenar y recuperar datos. Cada elemento se compone de una **clave, tiempo de caducidad y datos sin procesar**. En un Memcached de alto nivel funciona de la siguiente manera:

1. El cliente solicita un dato que Memcached verifica si está almacenado en el caché
2. Hay dos resultados posibles aquí:
   1. Si los datos se almacenan en caché: devuelva los datos desde Memcached (no es necesario verificar la base de datos)
   2. Si los datos no están almacenados en la memoria caché: consulte la base de datos, recupere los datos y luego guárdelos en Memcached
3. Cada vez que se cambia la información o el valor de caducidad de un elemento ha caducado, Memcached actualiza su caché para garantizar que se entrega contenido nuevo al cliente

Una configuración típica tiene varios servidores Memcached y muchos clientes. Los clientes usan un algoritmo de hashing para determinar qué servidor de almacenamiento de Memcached usar, esto ayuda a distribuir la carga. Luego, el servidor calcula un segundo hash de la clave para determinar dónde debe almacenar el valor correspondiente en una tabla de hash interna. Algunos puntos importantes sobre la arquitectura de Memcached incluyen:

* Los datos solo se envían a un servidor
* Los servidores no comparten datos
* Los servidores mantienen los valores en la RAM: si la RAM se agota, el valor más antiguo se descarta

## Arquitectura de Memcached



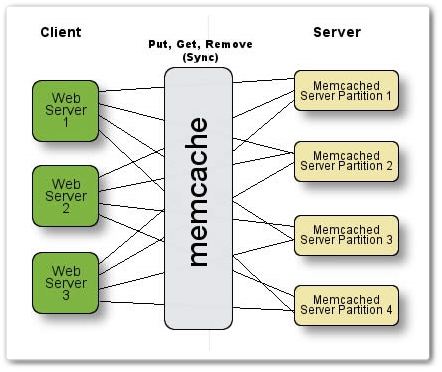
El sistema usa una arquitectura **cliente-servidor**. Los servidores mantienen un **array asociativo** clave-valor; los clientes añaden datos al array y acceden a él. Las claves pueden tener una longitud de hasta 250 bytes y los datos pueden tener un tamaño de hasta **1 megabyte**.

Los clientes usan librerías cliente para acceder a los servidores que, por defecto, utilizan el puerto 11211. Cada cliente mantiene una lista de todos los servidores; los servidores no se comunican entre ellos. Si un cliente desea establecer o leer el valor correspondiente a cierta clave, la librería cliente primero hace un cálculo mediante un algoritmo **hash** para determinar el servidor que va a utilizar. Entonces se pone en contacto con el servidor y éste usará otro hash para determinar dónde almacenar o leer el valor correspondiente.

El servidor mantiene los valores en RAM. Si un servidor agota su memoria, descarta los valores más antiguos. Por tanto, los clientes deben de tratar Memcached como una caché transitoria; no pueden asumir que los datos almacenados en Memcached estarán ahí cuando los necesiten. Un producto compatible a nivel de protocolo con Memcached llamado **MemcacheDB** proporciona almacenamiento permanente. Hay también una solución llamada **Membase** de Northscale que proporciona persistencia, replicación y clustering.

Para que un cliente pueda leer los datos almacenados por otro cliente, deberían ambos usar el mismo algoritmo hash para localizar los servidores.

Un despliegue típico tendría varios servidores y muchos clientes. Sin embargo, es posible usar Memcached en un único ordenador, actuando simultáneamente como cliente y servidor.



## Protocolo

La comunicación de clientes con servidor es muy simple, y basada en comandos. El protocolo Memcached implementa 3 comandos de almacenamiento, con pequeñas diferencias en su funcionamiento:

* SET: Actualiza el objeto si no existía anteriormente, o lo agrega en caso contrario.
* ADD: Agrega el objeto solo si no existe.
* REPLACE: Actualiza el objeto solo si existe.

Memcached permite controlar el tiempo de vida de un objeto, indicando el “tiempo de expiración” para el mismo, en el momento de realizar una operación de almacenamiento.

A su vez, posee un comando de recuperación: GET. Podemos eliminar un objeto mediante el comando DELETE. Además, el protocolo implementa comandos para recuperar estadísticas, vaciar el cache, utilizar algún tipo de compresión, entre otros.

## Seguridad y ataques

La mayor parte de los despliegues de Memcached se hacen en redes corporativas "de confianza", donde los clientes se pueden conectar libremente a cualquier servidor. Hay casos, sin embargo, en los que Memcached es desplegado en redes inseguras o en las que los administradores querrían ejercer un control en la forma que se conectan los cliente.

Para este propósito Memcached puede ser compilado con el soporte opcional de autentificación **SASL**. El soporte SASL requiere el uso del protocolo binario.

Los ciberdelincuentes han descubierto una forma de abusar de los servidores **Memcached** ampliamente utilizados para lanzar más de 51,000 ataques DDoS poderosos que su fuerza original, lo que podría resultar en la caída de los principales sitios web e infraestructura de Internet. El ataque aparentemente abusa de los servidores Memcached sin protección que tienen UDP habilitado para entregar ataques DDoS 51.200 veces su fuerza original, convirtiéndolo en el método de amplificación más prominente jamás utilizado en la naturaleza hasta el momento.

**GitHub** lidió hace pocas horas con lo que hasta la fecha ha sido el ataque DDoS más grande de la historia que se haya reportado, y lo más notable: **fueron capaces de resolver la situación en 10 minutos**.

Entre las 17:21 y las 17:30 UTC del 28 de febrero de 2018, **GitHub** fue capaz de identificar y mitigar un ataque DDoS que impactó a la plataforma con una cantidad monumental de tráfico: **1.35 Tbps (terabits por segundo) enviados a través de 126.9 millones de paquetes por segundo**.

GitHub acaba de sobrevivir el ataque DDoS más grande de la historia: <https://www.genbeta.com/actualidad/github-acaba-de-sobrevivir-el-ataque-ddos-mas-grande-de-la-historia>

Descubrén una técnica que puede mitigar los ataques DDoS de Memcached: <https://tecnonucleous.com/2018/03/08/descubren-una-tecnica-que-puede-mitigar-los-ataques-ddos-de-memcached/>

## Ventajas

**Tiempos de respuesta por debajo del milisegundo**

Memcached almacena todos sus datos en la memoria principal del servidor. A diferencia de otras bases de datos, como PostgreSQL, Cassandra y MongoDB, que almacenan la mayoría de sus datos en el disco o en tarjetas SSD, los almacenes de datos en la memoria no tienen que ir repetidamente a buscar la información en el disco. Esto les permite soportar una cantidad mucho mayor de operaciones y ofrecer tiempos de respuesta más rápidos. El resultado: un rendimiento fulgurante con una media de lectura y escritura de menos de un milisegundo que puede soportar millones de operaciones por segundo.

**Simplicidad y facilidad de uso**

Memcached está diseñado para ser simple y genérico, de modo que resulta un servicio eficaz y fácil de usar en el desarrollo de aplicaciones. Hay muchos clientes de código abierto disponibles para los desarrolladores de Memcached. Entre los lenguajes admitidos se encuentran Java, Python, PHP, C, C++, C#, JavaScript, Node.js, Ruby, Go y muchos otros.

**Escalabilidad**

La arquitectura multihilo y distribuida de Memcached hace que sea fácil de escalar. Puede dividir sus datos en varios nodos, lo que le permite añadir nuevos nodos a su clúster para aumentar la capacidad. Además, como Memcached es multihilo, puede usar varios núcleos en un determinado nodo. De este modo, se simplifica la escalación de la capacidad computacional. Con Memcached, puede crear soluciones de almacenamiento en caché distribuidas y escalables diseñadas para proporcionar un rendimiento constante y rápido.

**Comunidad**

Memcached es un proyecto completamente desarrollado de código abierto que se sostiene gracias a una comunidad dinámica. Aplicaciones como WordPress y Djando admiten el uso de Memcached para mejorar el rendimiento. No hay limitaciones de proveedores ni tecnología porque Memcached está basado en estándares abiertos, admite formatos de datos de código abierto y cuenta con una base de clientes completa.

## Inconvenientes

– Por cada clave solamente se permite almacenar 1 Mb de datos serializados. Esto es una auténtica burrada y cabe medio quijote pero según lo que queramos almacenar podemos pasarnos. Lo cachondo del caso es que no salta excepción sino que simplemente devuelve false.

– El servidor de memcached tiene definida una memoria de uso. Si cacheamos muchas cosas, se puede producir un overflow y las claves más antiguas empiezan a expirar para hacer sitio a las nuevas. Mucho cuidado pues con configurar correctamente esto.

– En alta escalabilidad, hasta que está todo cacheado puede darse el caso de muchas peticiones intentando cachear lo mismo simultáneamente. Este fenómeno es conocido como cache storming y aunque dura poco podemos tener problemas en aplicaciones donde de golpe la demanda aumenta drásticamente.

– No se pueden hacer querys para recuperar las keys almacenadas ni un rango de las mismas. Es decir, mucho cuidado con elegir las keys en los setData ya que nos podemos encontrar con no saber regenerarlas y por tanto no poder expirar keys concretas.

– Si no dimensionamos bien la aplicación, podemos estar haciendo muchos setData y pocos getData (porque el tiempo de expiración es muy bajo o porque cacheamos datos de poca consulta).

– Si confiamos en que memcached aguantará la carga y no optimizamos la aplicación, el día que se cae memcached, adios aplicación web en escasos minutos.

– Si los datos a cachear son muy grandes, la serialización de los mismos se vuelve muy costosa. Esto es especialmente grave con la vieja librería memcache y subsanado en parte con memcached+igbinary. Siempre irá más rápido que una query, especialmente si hacemos varias joins, pero tal vez la ganancia no sea tan grande como esperamos.

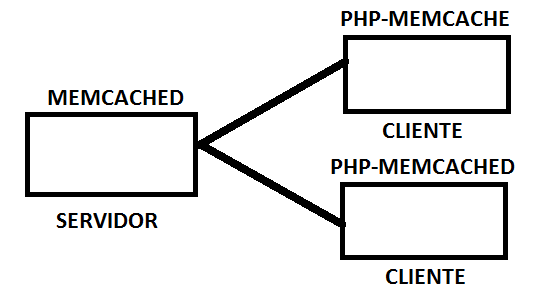
– Memcached no lleva autenticación. Es decir, cualquiera con acceso al servidor, si conoce las keys puede recuperar nuestros datos. Debido a esto, raramente está instalado en servidores compartidos.

## Datos curiosos

**Redis frente a Memcached**

Redis y Memcached son dos de los más conocimos almacenamientos de datos en memoria de valor clave. Memcached está diseñado para la simplicidad mientras que Redis ofrece un conjunto enriquecido de características que lo hacen efectivo para una amplia gama de casos de uso.

**Memcached Vs Memcache**



Como servidor solo existe uno: **Memcached**. Pero como **cliente para PHP** esta **MEMCACHED** Y**MEMCACHE**, cada uno con sus características.

La primera versión de **PHP-Memcache** apareció en el **año 2004**, **PHP-Memcached** apareció usando **PHP-Memcache** como base en el **año 2009**, por lo que **PHP-Memcache** está mucho más actualizado y por eso consigue **mayor velocidad de respuesta**.

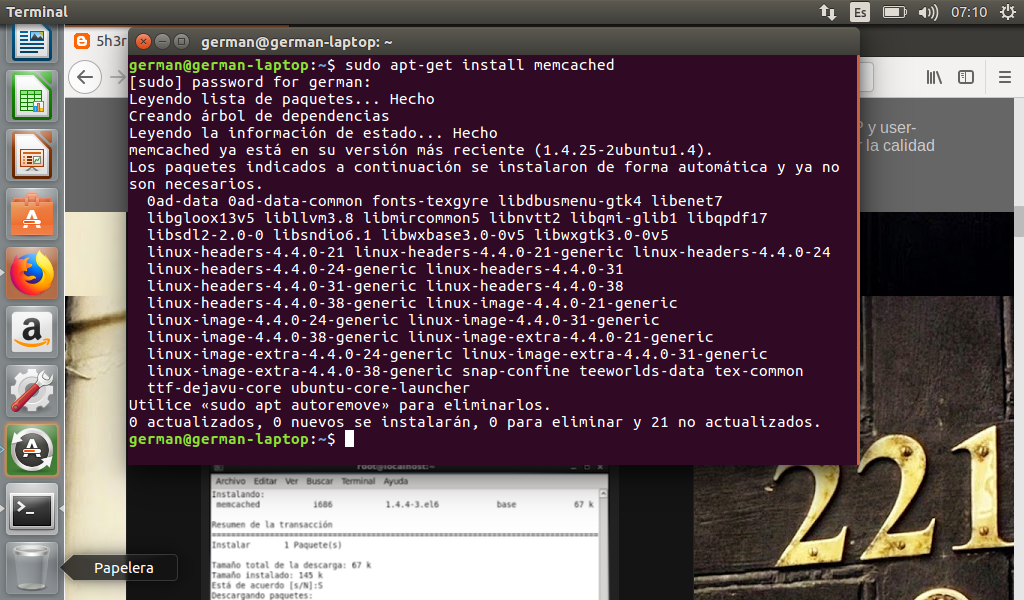
En esta tabla puedes ver**todas las diferencias técnicas que puedes encontrarte entre Memcache y Memcached**:

Pero resumiendo, según los **benchmarks** realizados por algunos de los sitios tecnológicos **PHP-Memcached es un 16% más rápido que PHP-Memcache**.

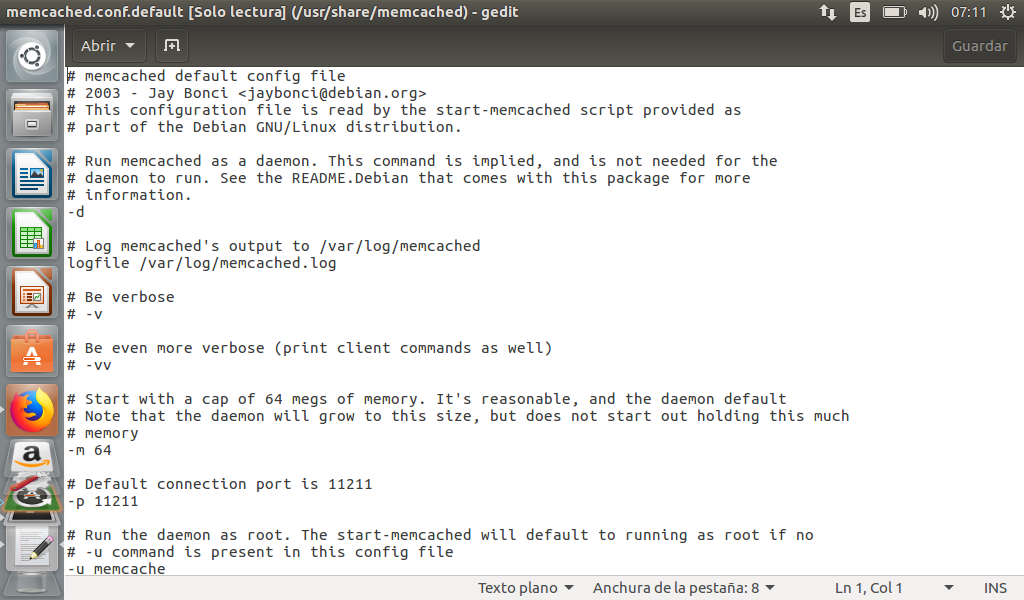
## Instalación

Abro una terminal y escribo lo siguiente:

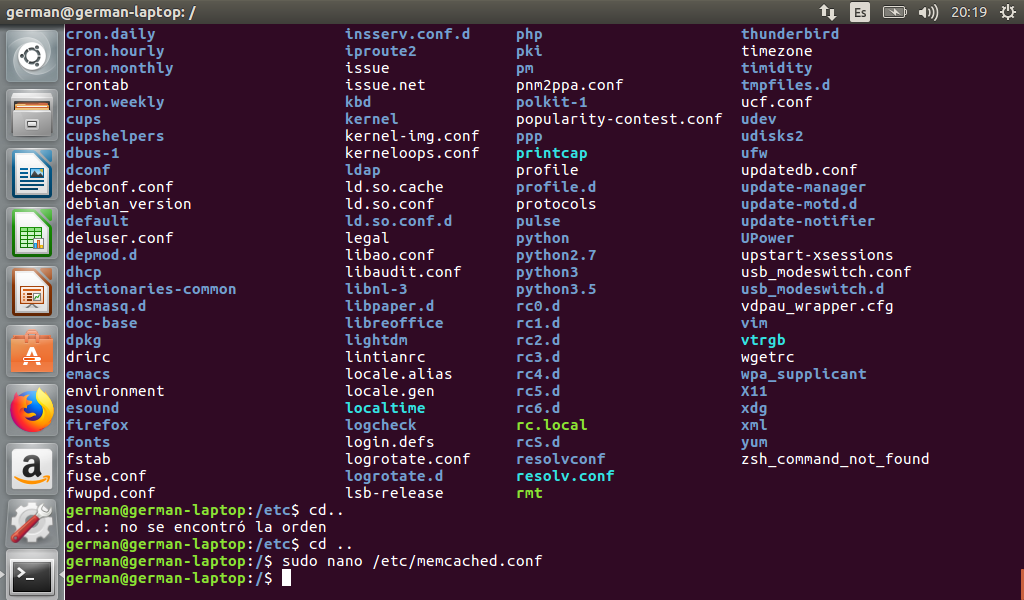
|  |  |
| --- | --- |
| 1  2  3 | sudo apt-get update    sudo apt-get install php-memcached memcached |

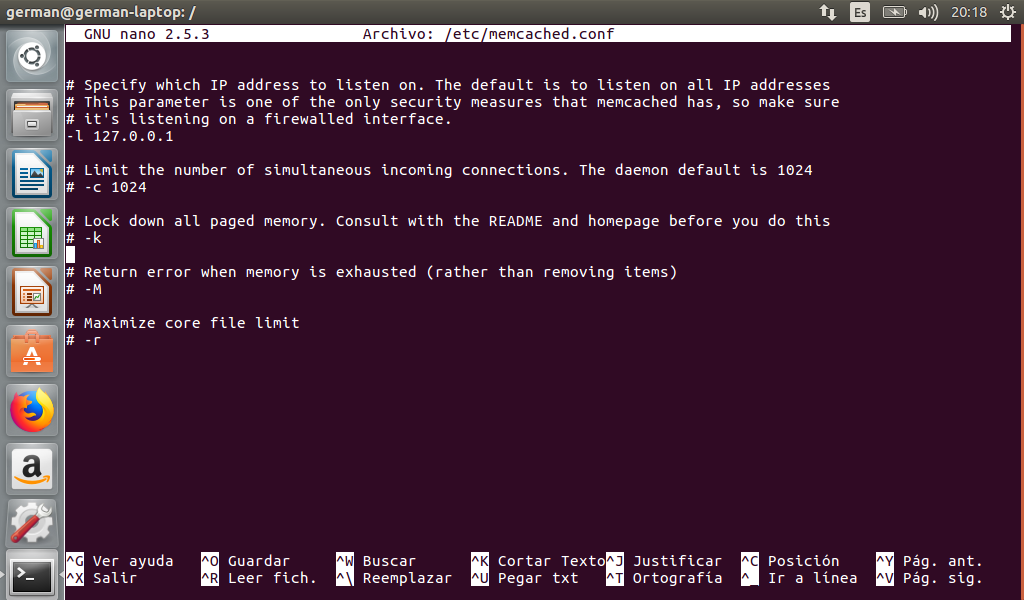


Compruebo las dependencias descargadas: vim /etc/memcached.conf (con editor vim)

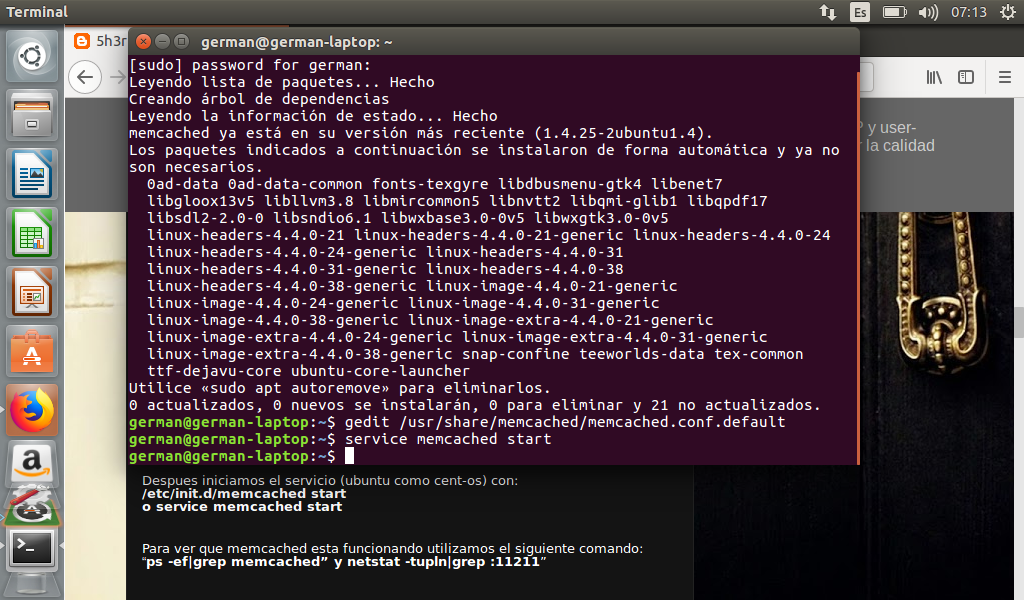


Con editor nano:

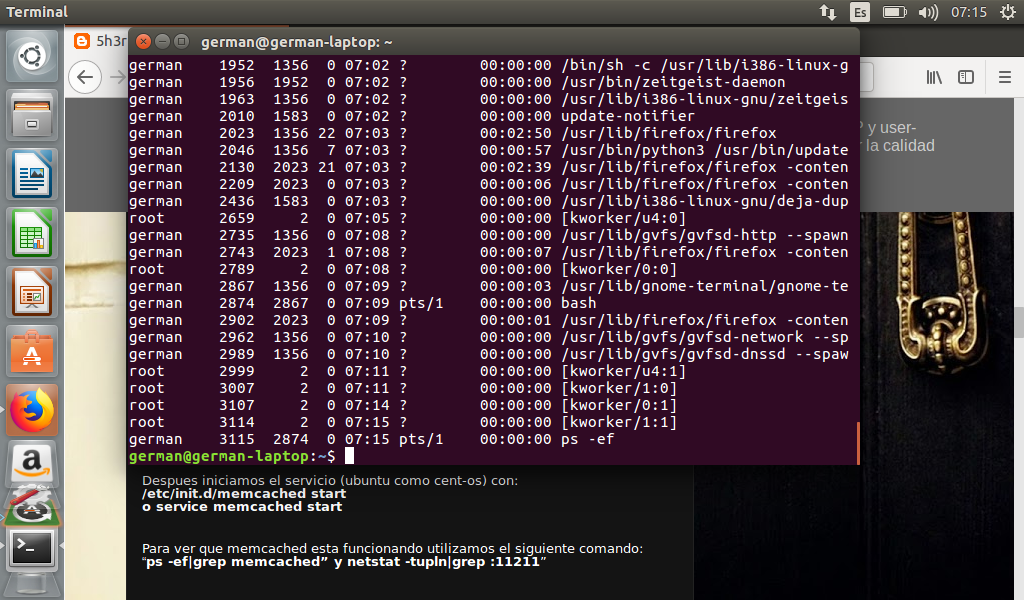




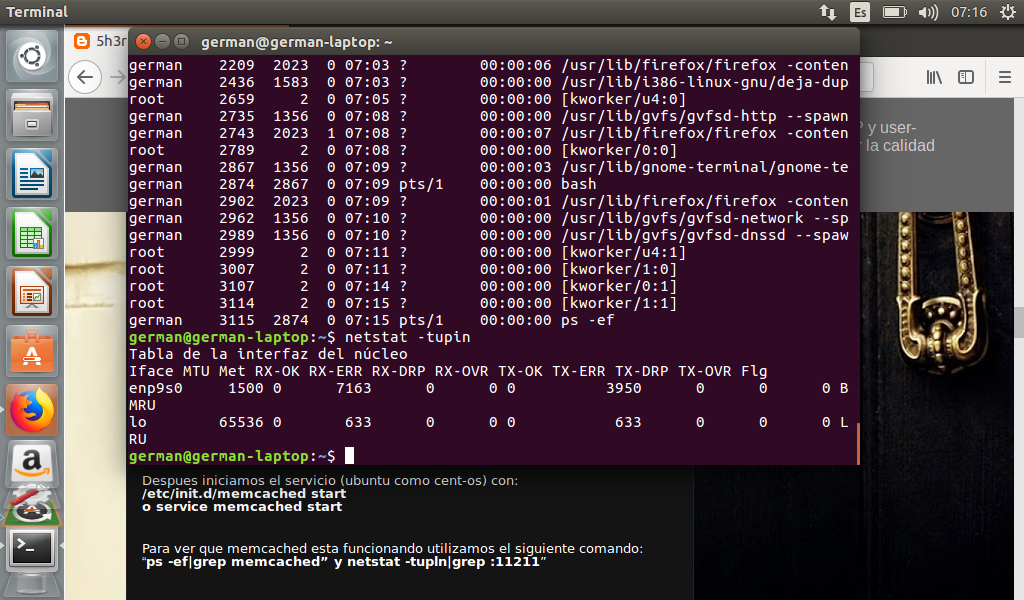
Reboto el Sistema Memcached: systemctl restart memcached



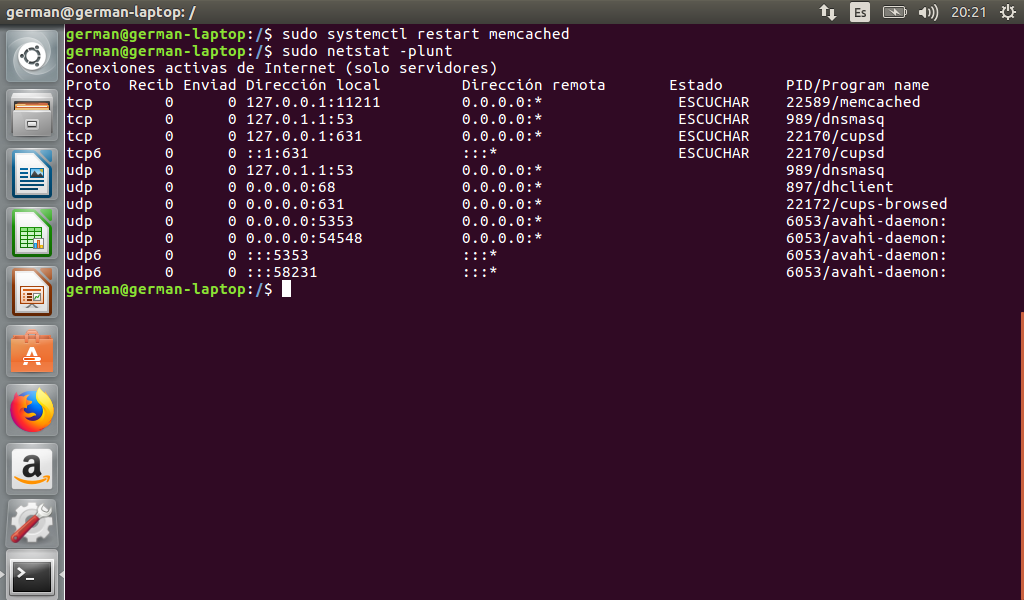
Compruebo los ajustes de estas configuraciones: ps –ef



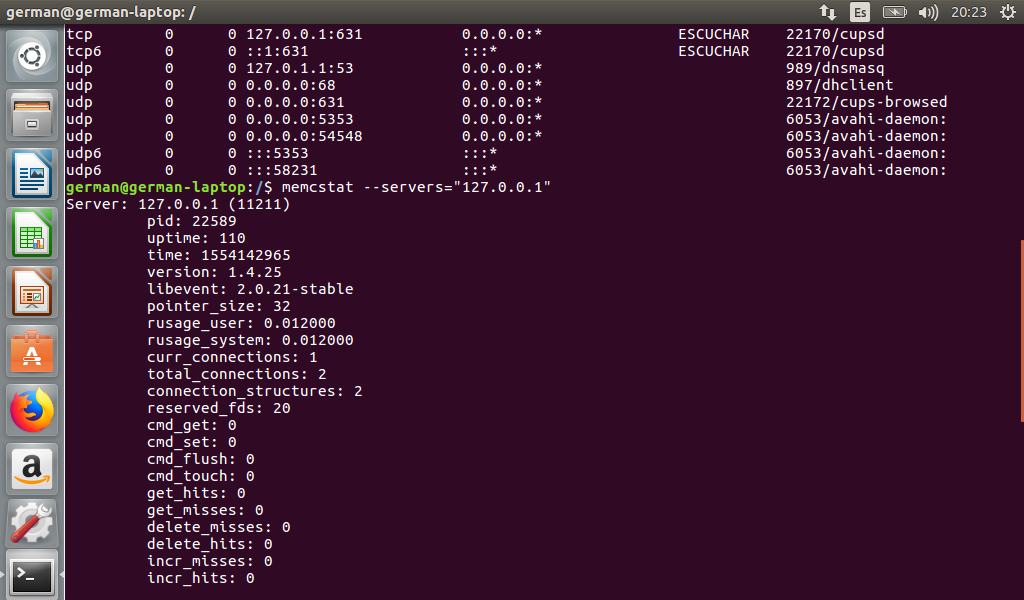
Con netstat –tupin:



Con netstat –plunt:



Comprobación de la ip( localhost):



Para comprobar el funcionamiento de Memcached creamos un archivo llamado info.php en la siguiente carpeta: **/var/www/html**. En este archivo introduzco las siguientes lineas de texto:

|  |  |
| --- | --- |
| 1 | <? ​phpphpinfo(); ​?> |

Una vez que hemos guardado el texto en el documento, nos dirigimos al navegador web y escribimos lo siguiente en la barra de navegación:

|  |  |
| --- | --- |
| 1 | localhost/info.php |

Y el resultado que me salió y nos debe salir es el siguiente:

