

# Técnicas Avanzadas en Gestión de Sistemas de Información

Facultad de Ingeniería de la Universidad de la República

## Arquitecturas en sistemas masivos de Redes Sociales

Docente: Prof. Hermann Steffen

### Grupo I

Gabriel Centurión 2.793.486-8  
Rodrigo Mendoza 5.292.491-1  
Mauricio Piñatares 4.407.245-3  
Santiago Maximo 4.647.623-3  
Sebastian Lorenzo 4.116.126-7  
Sebastian Cervantes 4.647.156-2  
Nicolas Granja 4.270.650-7

## **Tabla de contenido**

### **1. INTRODUCCIÓN**

#### **1.1 TEMA DE ESTUDIO**

#### **1.2 OBJETIVOS DEL ESTUDIO**

#### **1.3 RESUMEN**

### **2. CONTEXTO TECNOLÓGICO**

### **3. DESARROLLO DEL ESTUDIO**

#### **3.1 ASPECTOS METODOLÓGICOS**

#### **3.2 ESTUDIO REALIZADO: ASPECTOS TÉCNICOS Y CONCEPTUALES**

#### **3.3 IMPLEMENTACIÓN DE PRUEBA DE CONCEPTO**

### **4. CONCLUSIONES**

### **5. BIBLIOGRAFÍA**

### **6. ANEXOS**

# **1. Introducción**

El presente informe es el resultado del estudio realizado por el Grupo I en el marco de tareas previstas en la materia electiva Técnicas Avanzadas para la Gestión de Sistemas de Información de la carrera de Ingeniero en Computación, del Instituto de Computación de la Facultad de Ingeniería.

Hay muchos problemas inherentes al manejo de sistemas con millones de usuarios, como el tamaño de la base de datos requerida para mantenerlos y la cantidad de consultas realizadas a la misma. Además, por ser sistemas que apuntan a la interacción social, el nivel de interconexión entre los datos es muy alto, a esto se suma la cantidad de tráfico que generan los millones de usuarios hacia los servidores. Para resolver este tipo de problemas, se utilizan diversas soluciones arquitectónicas basadas en tecnologías como la utilización de granjas de servidores, la implementación de balanceo de carga, así como también el uso de tecnologías como Hadoop, TAO, BigPipe, entre otros.

## **1.1 Tema de estudio**

El tema a estudiar es la escalabilidad de las arquitecturas de redes sociales, como sistemas masivos distribuidos. Las populares redes sociales presentan infraestructuras significativamente desafiantes. Cientos de millones de personas usan sus redes cada día e imponen una demanda computacional que las arquitecturas tradicionales luchan por satisfacer. La infraestructura de las redes sociales deben permitir comunicación en tiempo real, obtener el contenido de múltiples fuentes en la marcha, ser capaz de acceder y actualizar contenido compartido muy popular y escalar para poder procesar millones de solicitudes de los usuarios por segundo.

En este informe estudiaremos como las redes sociales más populares hacen frente a estos desafíos, haciendo enfoque en su arquitectura general, la organización en capas, granja de servidores, base de datos distribuidas, técnicas de optimización y performance de sus sitios web, mecanismos de distribución de carga, de respaldo y recuperación, procesamiento de grandes volúmenes de datos, etc.

## **1.2 Objetivos del Estudio**

Los principales objetivos de este estudio han sido:

1. Entender las claves de la arquitectura de los sistemas masivos distribuidos, teniendo como referencia los casos de las redes sociales.
2. Estudiar en detalle las distintas tecnologías utilizadas para implementar la arquitectura de forma de soportar las exigencias que éstos tipos de sistemas tienen.
3. Evaluar la performance de cada una de las partes de esta arquitectura.
4. Analizar la viabilidad de integrar las tecnologías planteadas con otras ya existentes en el mercado y con futuras implementaciones.

### 1.3 Resumen

En este informe se realiza un estudio de cómo se resuelven los problemas más importantes en sistemas de gran porte como son las redes sociales.

La distribución de los datos y el acceso a los mismos es una cuestión fundamental. Para lograr eficiencia, básicamente en lecturas, Facebook diseñó un data store geográficamente distribuido: TAO. El mismo provee un acceso eficiente al grafo social ofreciendo un conjunto de consultas que luego se mapean a MySQL (capa de almacenamiento). Para ello implementa una capa de cache con datos actualizados, utilizando un esquema de líderes y seguidores para poder escalar a muchos servidores, y sobre éste un diseño maestro/esclavo para reducir la latencia de red, todo lo cual lleva a que Facebook sea una red social eventualmente consistente.

Diversos estudios han demostrado que los usuarios tienden a abandonar sitios en los cuales los tiempos de respuesta son demasiado extensos. Como consecuencia, Facebook busca continuamente soluciones que minimicen los tiempos de respuesta, no solo al momento de diseñar sus data stores y granjas, sino que inclusive optimiza los tiempos de renderizado. Para esto último utiliza librerías específicas como Primer, y técnicas como BigPipe.

Las redes sociales necesitan de múltiples servidores conectados entre sí para poder responder a sus millones de solicitudes. Para alcanzar ciertos tiempos de respuesta y evitar que sus servidores se saturen, se utilizan métodos de balanceo de carga dentro de las granjas. Estos métodos permiten que los pedidos de los usuarios sean atendidos por el servidor más cercano a su origen, de forma de disminuir la latencia. Y además permite reaccionar ante el caso de que exista una falla en la red, tanto en los servidores como en los ISP, encontrando una forma de solucionar la falla.

Las grandes empresas, y en particular, las redes sociales, manejan inmensas cantidades de información por día, la cual se torna compleja de administrar y consultar. Las redes sociales son parte de esta realidad por lo que utilizan una serie de tecnologías para el manejo de grandes volúmenes de datos. En el siguiente documento presentamos algunas de las tecnologías utilizadas por redes sociales, sus pormenores y exponemos un caso de estudio de un sitio de retail.

Entre las tecnologías presentadas se encuentra Hadoop, sistema que nos permite administrar el sistema distribuido para administrar los grandes volúmenes de datos. Además indicamos que tecnologías componen la plataforma como interactúan entre sí como ser hive para realizar consultas, el sistema de archivos distribuido hdfs, la base de datos distribuida hbase entre otras.

## 2. Contexto tecnológico

### Redes sociales.

Las redes sociales en Internet son comunidades virtuales donde sus usuarios interactúan con personas de todo el mundo con quienes encuentran gustos o intereses en común. Funcionan como una plataforma de comunicaciones que permite conectar gente que se conoce o que desea conocerse, y que les permite centralizar recursos, como fotos y vídeos, en un lugar fácil de acceder y administrado por los usuarios mismos.

Las redes sociales en internet se basan en los vínculos que hay entre sus usuarios. Existen varios tipos de redes sociales:

- **Redes sociales genéricas.** Son las más numerosas y conocidas. Las más extendidas en España son Facebook, Tuenti, Google+, Twitter o Myspace.
- **Redes sociales profesionales.** Sus miembros están relacionados laboralmente. Pueden servir para conectar compañeros o para la búsqueda de trabajo. Las más conocidas son LinkedIn, Xing y Viadeo.
- **Redes sociales verticales o temáticas.** Están basadas en un tema concreto. Pueden relacionar personas con el mismo hobby, la misma actividad o el mismo rol. La más famosa es Flickr.

La primera red social disponible en internet es una pagina llamada Classmates.com creada en 1995. Pero la mas popular es Facebook que fue creada en 2004.

Durante el ultimo año el crecimiento de las redes sociales ha sido notable, si tomamos en cuenta que cada vez son más los que se agregan y muchos otros que las utilizan con más reincidencia, siendo actualmente Facebook la mas popular y seguida de lejos por Twitter. A continuación mostramos la cantidad de usuarios activos de las redes sociales mas populares de la actualidad [1]:

1. Facebook: 1.1 mil millones de usuarios
2. Twitter: 560 millones de usuarios
3. Google+: 400 millones de usuarios
4. LinkedIn: 240 millones de usuarios
5. Instagram: 150 millones de usuarios
6. Pinterest: 70 millones de usuarios

### **3. Desarrollo del Estudio**

#### **3.1 Aspectos metodológicos**

Lo primero que se hizo fue buscar información sobre arquitectura de sistemas masivos distribuidos, recolectar los principales puntos de la temática y dividirlos entre los integrantes. Cada uno de ellos se informó en su tema en general y lo enfocó en la temática principal del informe. Los temas estudiados por los distintos integrantes del grupo son temas que están muy conectados entre sí, por lo cual en la búsqueda de información de cada uno de ellos, muchas veces se debió interiorizarse en los demás. Esto fue muchas veces una dificultad, ya que no fue fácil encontrar información específica de cada uno de ellos.

Como todo aspecto tecnológico, ocurren cambios y mejoras de forma constante, por lo cual es difícil validar que la información tratada esté actualizada. Teniendo en cuenta que en el periodo de un año se encuentran publicaciones que al día de hoy ya son obsoletas.

No fue fácil identificar entre la información encontrada cuál hacía referencia a datos verídicos y no a especulaciones o teorías de cómo podrían funcionar idealmente las redes sociales.

#### **3.2 Estudio realizado: aspectos técnicos y conceptuales**

Mike Schroepfer, vicepresidente de ingeniería de Facebook, hizo recientemente una entrevista en la que dijo: "La ampliación de cualquier sitio web es un reto, pero la ampliación de una red social, tiene retos únicos." Él continuó diciendo que a diferencia de otros sitios web, no se puede simplemente añadir más servidores para resolver el problema debido al "enorme conjunto de datos interconectados." Nuevas conexiones se crean todo el tiempo debido a la actividad del usuario.

Facebook ha crecido tan rápidamente que a menudo se enfrentan con cuestiones relativas a las consultas de base de datos, el almacenamiento en caché y el almacenamiento de datos. Su base de datos es enorme y en gran medida compleja. Para dar cuenta de esto, Facebook ha comenzado una gran cantidad de proyectos de código abierto y servicios de back-end, muchos de los cuales nombraremos mas adelante, ya que han ayudado en gran medida a resolver gran cantidad de los retos únicos que nombraba Mike Schroepfer.

## Granja de servidores.

Una granja de servidores, también llamada cluster de computadoras, es una colección de servidores en red que trabajan juntos para proveer recursos del lado del servidor. Se pueden agrupar múltiples servidores en una granja de servidores para representar una misma entidad o recurso, como un web server o un servidor de aplicaciones específico. En esas situaciones, se puede configurar la granja de servidores para distribuir requests entre los servidores del clúster de acuerdo a sus cargas de trabajo y disponibilidad.

Se pueden usar granjas de servidores para proporcionar un grado de redundancia para las funciones específicas del servidor. Por ejemplo, si un servidor falla, la granja de servidores puede automáticamente dirigir requests a un segundo servidor.

Las granjas de servidores también pueden proporcionar una alternativa más escalable y flexible para la actualización de servidores existentes mediante la adición de hardware. [2]

El balanceo de carga y evitar los cuellos de botella son elementos críticos para determinar la escalabilidad de una granja. El balanceo de carga se lleva a cabo por medio de dispositivos externos encargados de distribuir la carga del trabajo o el tráfico de la red a través de todos los servidores que constituyen la granja. Los servidores están interconectados directa o indirectamente al dispositivo de balanceo que realiza la distribución de la carga. Estos utilizan algoritmos para decidir qué servidor recibirá la próxima petición entrante. [3]

Las granjas de servidores son fundamentales en la arquitectura de redes sociales dado su manejo de múltiples requerimientos de forma distribuida y su alta disponibilidad, ya que requieren el manejo de una enorme cantidad de pedidos por segundo, con tiempos de respuesta muy rápidos (2,47 segundos de promedio para facebook) y altísima disponibilidad (97,8 % para el caso de facebook, en febrero de 2013). [4]



Granja de servidores de facebook. [5]

## Distribución de datos en Facebook: TAO

TAO es un data store geográficamente distribuído, especializado en lecturas, que provee un acceso eficiente al Grafo Social de Facebook a través de un conjunto de consultas predefinidas. Es desplegado en facebook, reemplazando a memcache. Se ejecuta en miles de servidores, y puede procesar un mil millones de lecturas y un millón de escrituras por minuto.

Una simple página de Facebook manipula cientos de items del Grafo Social, que se exponen de acuerdo al usuario y a su privacidad, lo que implica una exigente demanda de lecturas.

Facebook originalmente fue construido almacenando el Grafo Social en MySQL (TAO continúa usando MySQL como base de datos), consultándolo a través de PHP y “cacheando” resultados en Memcache. Más adelante se creó una abstracción en PHP para leer y escribir sobre el grafo, evitando el acceso directo a MySQL. Luego surge TAO, un servicio que implementa directamente los objetos y asociaciones del modelo, permitiendo fácilmente el acceso al grafo por parte de servicios no implementados en PHP, y cubriendo falencias de la API de PHP y de Memcache.

### Conceptos Previos: Memcache y Grafo Social

Memcache es un sistema de almacenamiento en memoria cache distribuido. Usualmente se utiliza para acelerar sitios web dinámicos con bases de datos almacenando en caché datos y objetos en memoria RAM para reducir el número de veces que una fuente de datos externa (como una base de datos o API) debe ser leído. La API de Memcache provee una tabla hash distribuida en memoria sobre un cluster de computadoras, y se utiliza para proporcionar acceso de baja latencia a un pool de almacenamiento compartido a bajo costo.

Se llama Grafo Social al mapa global de las personas que forman una red social y de las relaciones entre ellas. Facebook da un paso más allá: integra en el grafo social otras cosas además de las personas. ¿Y cómo se integran en el grafo social todos los elementos de Facebook que no corresponden a personas físicas (páginas, grupos, eventos...)? Facebook trata tanto a las personas como al resto de entidades como “objetos” del grafo. [6]

### El modelo de datos de TAO

El objetivo de TAO no es soportar el conjunto completo de consultas al grafo, pero sí proveer suficiente expresividad para manejar los principales requerimientos y al mismo tiempo permitir una implementación escalable y eficiente.

Los objetos en TAO son los nodos del grafo y las asociaciones las aristas. Los objetos están identificados con un entero de 64 bits que es único entre todos los objetos. Las asociaciones están identificadas por el identificador del objeto fuente, el tipo de asociación y el identificador del objeto destino. Solo puede existir una asociación de un determinado tipo entre 2 objetos. Las asociaciones representan acciones que pueden suceder una sola vez o transiciones de estados, como la aceptación a la invitación de un evento. Las acciones repetitivas son mejor representadas a través de objetos. A pesar de que las asociaciones son dirigidas, comúnmente están acopladas a una arista inversa.

## **La API de TAO**

La API de objetos de TAO provee operaciones para asignar un nuevo objeto, recuperar, actualizar o borrar un objeto asociado a un identificador.

Las aristas bidireccionales son representadas mediante 2 asociaciones separadas. TAO provee soporte para mantener sincronizada una asociación con su inversa, a través del tipo inverso. Para este tipo de asociaciones las creaciones, updates y deletes son automáticamente acopladas con una operación en la asociación inversa.

En TAO las consultas sobre asociaciones se organizan en torno a lista de asociaciones (listas de todas las asociaciones entre un id particular y un tipo de asociación, ordenadas en orden descendiente de acuerdo al tiempo) permitiendo consultar por ejemplo por los 100 comentarios más recientes sobre una publicación.

## **La Arquitectura de TAO**

TAO está separada en 2 capas, una de capa de cache y otra de almacenamiento.

### **Capa de almacenamiento**

La API de TAO está mapeada a un conjunto de consultas SQL, pero puede ser mapeada eficientemente a otros sistemas de bases de datos que no estén basados en dicho lenguaje.

Debido a la necesidad de TAO de manejar grandes cantidades de datos, estos se dividen en fragmentos lógicos: shards. Cada shard está contenido en una base de datos lógica, y cada servidor de base de datos puede contener uno o más shards. Por defecto los objetos son almacenados en una tabla y las asociaciones en otra. Los objetos son asociados a un shard durante todo su ciclo de vida, y las asociaciones son almacenadas en el shard de su objeto origen y por lo tanto toda consulta relacionada a una asociación va a ser resuelta a través de un solo servidor.

### **Capa de cache**

La capa de cache consiste en múltiples servidores de cache que en conjunto forman un tier. Un tier es colectivamente capaz de responder a cualquier consulta de TAO. Cada consulta es mapeada a un único servidor de cache, en un esquema de shards similar al descrito anteriormente. Los clientes dirigen sus solicitudes a un servidor de cache en particular, el cual es el responsable de completar la lectura o escritura correspondiente. En caso de fallas de lecturas o en el caso de escrituras, el servidor se comunica con otros servidores y/o bases de datos.

La cache de TAO contiene objetos, lista de asociaciones y recuento de asociaciones. La cache se “llena” a demanda y se van eliminando datos de acuerdo al mecanismo de LRU (least recently used). Los servidores cache entienden la semántica de sus contenidos y las utilizan para procesar consultas que incluso no habían sido exactamente procesadas de forma previa (por ejemplo un recuento cacheado en cero es suficiente para responder una consulta de rango).

Las operaciones de escritura sobre una asociación con una inversa pueden involucrar 2 shards, asociados a los identificadores id1 (objeto origen) e id2 (objeto destino). El tier que recibe dicha solicitud envía una llamada RPC al miembro que almacena el id2, quien se contacta con la base de datos para realizar la escritura. Una vez que la escritura inversa es completada, el servidor de cache envía la escritura a la base de datos del id1. TAO no provee atomicidad entre los 2 updates. Si una falla ocurre, la primera asociación creada no tendrá inversa, estas asociaciones pendientes son programadas para ser reparadas en procesos asincrónicos.

### **Líderes y seguidores**

En teoría un solo tier podría ser escalado para manejar cualquier cantidad de solicitudes, teniendo en cuenta la división por shards. Pero en la práctica a medida que crecen son propensos a un alto crecimiento en la cantidad de conexiones.

Para agregar nuevos servidores pero limitando el tamaño máximo de un tier, se divide al cache en dos niveles: un tier líder y muchos tiers seguidores. Cada tier contiene un conjunto de servidores cache que juntos pueden responder cualquier consulta de TAO, es decir cada shard en el sistema se mapea a un servidor de cache en cada tier. Los líderes actúan igual que en el caso de un tier único, escribiendo y leyendo sobre la capa de almacenamiento. Los seguidores en cambio enviarán las lecturas con fallas y las escrituras a su líder. Los clientes se contactan con el seguidor más cercano disponible y nunca se comunican con los líderes.

Cada shard es mantenido por un líder, y todas las escrituras a ese shard son dirigidas a través de ese tier, por lo tanto es naturalmente consistente. Los seguidores en cambio deben ser notificados de los cambios hechos por otros seguidores. TAO

provee consistencia eventual, enviando de forma asíncrona mensajes de mantenimiento desde el líder a los seguidores. El seguidor que envió la solicitud de escritura es actualizado sincrónicamente en la respuesta del líder, ignorando el posterior mensaje asíncrono asociado (a través de un número de versión).

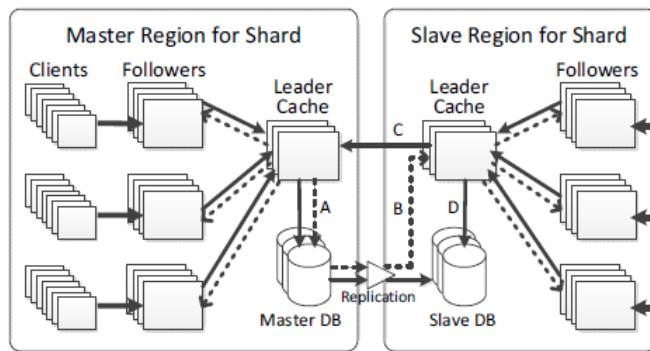
Como los líderes manejan todas las solicitudes relacionadas a un id, serializan las escrituras concurrentes y también establecen un máximo de solicitudes encoladas por shard.

### Escalabilidad a través de maestros y esclavos

El esquema de líderes y seguidores permite escalar para manejar grandes cantidades de carga, pero bajo la suposición de que la latencia de red entre los seguidores y los líderes, y entre éstos y las bases de datos, son bajas, algo que no sucede en realidad ya que no todos los centros de datos son próximos.

Actualmente los tiers seguidores pueden estar separados por miles de kilómetros, y en este contexto la latencia puede transformarse en un cuello de botella. Como las fallas de lecturas son 25 veces más frecuentes que las escrituras, se optó por una arquitectura maestro/esclavo que requiere que las escrituras sean enviadas al maestro, pero permite que las fallas de lectura sean resueltas localmente. Como en el diseño líder/seguidor, se propagan las notificaciones de actualizaciones asíncronamente para maximizar la performance y disponibilidad, a expensas de la frescura de los datos.

Debido a la actividad de la red social, es imposible agrupar usuarios de forma tal que las solicitudes no crucen diferentes particiones, por lo tanto cada seguidor va a ser local a una tier de base de datos que contenga todo el grafo social. Los centros de datos están agrupados en unas pocas regiones, dentro de las cuales hay una baja latencia.



La región maestra envía fallas de lecturas, escrituras y mensajes de consistencia embebidos, a la base de datos maestra (A). Los mensajes de consistencia son enviados al líder esclavo (B) a medida que el stream de replicación actualiza la base de datos esclava. El líder esclavo envía escrituras al líder maestro (C) y las fallas de lectura a la base de datos replicada (D). La elección del maestro y el esclavo es separada para cada shard.

Los seguidores se comportan de igual forma en todas las regiones, redirigiendo las fallas de lecturas y las escrituras al tier líder de la región. Estas lecturas son redirigidas a la base de datos local de la región, independientemente de si es una región maestra o esclava. Las escrituras, en cambio, son redirigidas por el líder local al líder de una región maestra. Esto significa que la latencia de lectura es independiente de la latencia entre regiones. Si una escritura es exitosa entonces el líder local (de la región esclava) actualiza su cache con el nuevo valor, aún cuando la base de datos esclava quizás no haya sido actualizada por el stream de replicación asincrónica.

El diseño maestro/esclavo asegura que todas las lecturas sean satisfechas en una única región a expensas de que puedan retornarse datos “viejos”. Si un cliente constantemente consulta al mismo seguidor, probablemente tenga una vista consistente del estado de TAO.

## Implementación

Detalles para la optimización de performance y la eficiencia de almacenamiento.

### Servidores cache

Facebook partitiona la memoria RAM existente en “arenas”, seleccionando la arena de acuerdo al tipo de objeto o asociación. Esto permite extender el tiempo de vida para tipos importantes.

Para items de tamaño pequeño y fijo, como recuentos de asociaciones, los punteros se tornan ineficientes y por lo tanto se los almacena directamente sin necesidad de punteros.

### Mapeo SQL

Cada shard es asignada a una base de datos lógica que tiene una tabla para objetos y una para asociaciones. Todos los campos de un objeto son serializados en una única columna de datos, lo que permite tener diferentes tipos de objetos en una única tabla. Los objetos que se benefician de tener una administración separada, se almacenan en distintas tablas.

Las asociaciones se almacenan de forma similar pero incluyen en el índice el tiempo para soportar consultas de rango. Para evitar consultas costosas de SELECT COUNT, los recuentos de asociaciones son almacenados en tablas separadas.

### Balanceo de cache y objetos populares

Al ser las shard mapeadas a un servidor de cache de forma relativamente aleatoria,

se puede producir un desbalanceo de carga y por lo tanto algunos seguidores van a lidiar con más solicitudes que otros. TAO rebalancea la carga a través de la clonación de shards, y de esta forma algunas lecturas son resueltas por múltiples seguidores en una misma región. Los mensajes de consistencia son enviados a todos los seguidores que contengan la shard.

Debido la magnitud de consultas sobre algunos objetos populares, también se los cachea del lado del cliente de acuerdo al valor de la tasa de accesos de los mismos, almacenando un número de versión. Las números de versiones también son utilizados para que un seguidor reconozca una copia antigua de una asociación, en el caso de que llegue una actualización de una versión más nueva.

## **Consistencia**

TAO es eventualmente consistente, después de una escritura garantiza transmitir el resultado a todos los tiers en un determinado período de tiempo. El retraso de replicación es usualmente menor a un segundo.

TAO provee consistencia de lectura después de escritura en una única tier, ya que actualiza sincrónicamente el servidor cache local luego de que la lectura ha sido exitosa (si es una asociación inversa actualiza ambos lados).

TAO no ofrece una fuerte consistencia, pero como se escribe en MySQL sincrónicamente, la base de datos maestra sí es un recurso consistente. Esto permite proveer un conjunto de solicitudes críticas, por ejemplo de autenticación, que son redirigidas a la región maestra.

## **Manejo da fallas**

TAO utiliza timeouts para dejar de esperar respuestas que quizá nunca lleguen y para determinar si ciertos servidores se encuentran “caídos”. También se consulta activamente por la recuperación de estos servidores.

Si una base de datos maestra no se encuentra disponible, automáticamente un esclavo pasa a ser el maestro.

Cuando una base de datos esclava no se encuentra disponible, las fallas de escritura son enviadas al líder maestro. Cuando la base vuelva a estar disponible se le enviarán los mensajes de actualización.

Cuando un servidor cache del líder falla, los seguidores redirigen las fallas de lectura directamente a la base de datos, las escrituras en la cache de ese líder son redirigidas a algún otro miembro (servidor) de la tier.

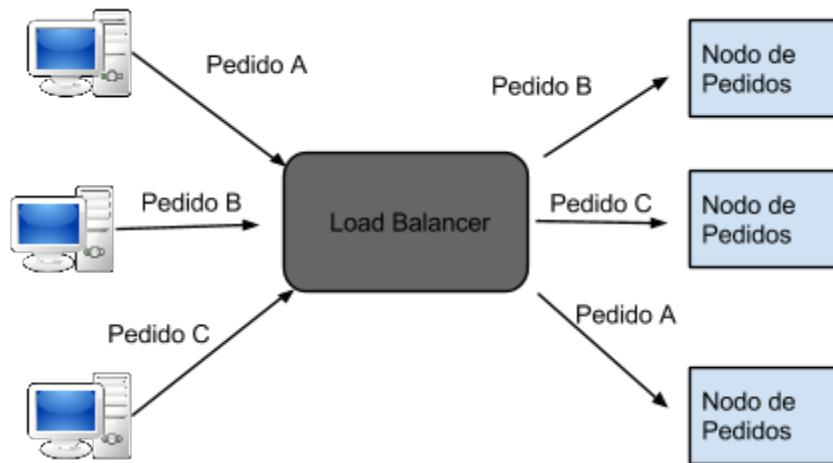
Si un seguidor falla, otro seguidor pasa a ser responsable por sus solicitudes. Cada cliente tiene un seguidor primario y uno de respaldo.

### **TAO en producción**

Facebook tiene una sola instancia de TAO en producción (Multi-tenancy). El sistema TAO contiene muchos tiers seguidores dispersos en diferentes regiones geográficas. Cada región tiene un juego completo de base de datos, un tier líder y al menos 2 seguidores. [7]

## Mecanismos de distribucion de carga.

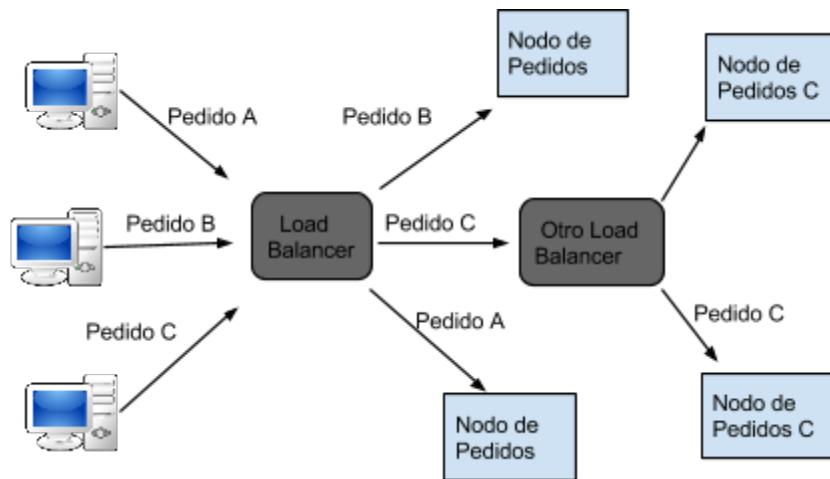
La distribución de carga es una pieza critica dentro de cualquier sistema distribuido. Esta tiene como principal rol repartir la carga entre un conjunto de nodos responsables de atender las peticiones. Esto permite que varios nodos atiendan pedidos de forma transparente un mismo pedido.



Existen muchos algoritmos diferentes que pueden ser usados para atender pedidos, incluyendo la elección de un nodo al azar, el algoritmo de round robin, o incluso eligiendo un nodo basado en cierto criterio, como podría ser la memoria o el uso del procesador.

Los平衡adores de carga pueden ser implementados como software o como dispositivos de hardware.

En un sistema distribuido, los balanceadores de cargas suelen ser encontrados al frente del sistema, de modo que todos los pedidos son enrutados como corresponde. En un sistema más complejo, es común que el pedido sea enrutado entre varios balanceadores de carga.



En esencia, estos dispositivos presentan una dirección de "servidor virtual" al mundo exterior, y cuando un usuario intenta conectarse, se direccionaría la conexión al servidor real mas apropiado, haciendo una traducción de forma bidireccional, tal y como se llevaría a cabo con un router que implementa NAT (Network Adress Translation).

El aspecto del balanceo de carga es bastante importante debido a que en muchas aplicaciones paralelas, como la búsqueda o la optimización, es extraordinariamente difícil predecir el tamaño de las tareas asignadas a cada procesador, de manera que se realice una división de las mismas para que todos mantengan la carga computacional uniforme. Cuando no es uniforme, es decir, hay desbalanceo en la carga, entonces algunos procesadores terminarán permaneciendo inactivos mientras otros todavía están calculando.

El estudio del balanceo de carga es muy importante para poder distribuir de una forma equitativa la carga computacional entre todos los procesadores disponibles y con ello conseguir la máxima velocidad de ejecución.

Las redes sociales, y en particular Facebook es usado por personas de todos los lugares del mundo, por lo tanto es necesario balancear ese tráfico con el objetivo de lograr la mayor performance posible.

Facebook tiene alrededor de un 1100 millones de usuarios activos, los cuales generan aproximadamente 12 millones de pedidos HTTP por segundo. Para manejar la carga, el tráfico es primero dirigido a unos pocos TCP proxies, estos luego reenvían los pedidos HTTP a uno gran grupo de HTTP proxies, quienes a su vez los mandan a hacia los servidores.

Facebook ha construido varios sistemas para manejar y balancear su tráfico, esto

incluye DNS load balancer y software de balanceo de carga capaz de manejar varios protocolos.

La distribución geográfica de estos usuarios es tal que la primera acción para el balanceo de carga es manejada por un servidor DNS. Tener una configuración DNS desactualizada significa el envío de una gran cantidad de pedidos HTTP de los usuarios hacia centros de datos que no están de manera óptima cercanos a ellos. Y esto se traduce en que las solicitudes sean más lentas.

El DNS load balancer tiene dos componentes principales: un motor de decisión GLB central escrito en Python, este realiza la tarea de tomar las decisiones del balanceo del tráfico y generar el mapeo DNS, el otro componente es un DNS server open source llamado TinyDNS, el cual dirige a los usuarios hacia los clusters basándose en una tabla cagada desde el mapeo DNS.

El motor de decisión GLB que usan se llama Cartographer. Este obtiene información como topología de internet, latencia y ancho de banda del usuario, datos de cluster de computo (carga/disponibilidad/performance), y a partir de estos y determina el mejor cluster disponible actualmente para cada usuario. Cartographer también recibe continuamente actualizaciones desde diferentes canales de monitorización agregando automáticamente nuevos mapeos DNS al servidor DNS cada vez que se necesita ajustar la carga de un cluster o reaccionar a problemas de red (se puede reaccionar a interrupciones de servicios dados por problemas en los clusters de Facebook o por problemas localizados en un país determinado o un ISP determinado).

## Mecanismos de respaldo y recuperación.

### Como se deberían manejar los Backups & Recoverys en los Data-Centers? [10]

Los respaldos y recuperaciones son casi de las mas importantes tareas que lleva cabo un administrador, y en el caso de los data-centers la situación no es diferente. De todas formas, debido al gran tamaño de la base de datos, los data-centers crean nuevos desafíos para los administradores en el área de los respaldos y restauraciones.

Los data-centers son únicos en el sentido en que los datos pueden venir de muchos recursos diferentes y ser transformados antes de ser finalmente insertados en la base de datos; igualmente sobre todo son únicos porque pueden ser realmente grandes. Administrar la recuperación de un gran data-center puede ser una tarea titánica, y el tradicional método de backup y recovery OLTP puede no ser suficiente para las necesidades del mismo.

Los data-centers difieren de los sistemas OLTP en lo siguientes puntos:

- Los data-centers son típicamente mucho mas grandes
- Un data-center puede llegar a tener muy diferentes requerimientos de disponibilidad en comparación a un sistema operacional. Más allá de que las decisiones empresariales dependen de la información del data-center, una situación en la cual un servicio no pueda operar es mucho peor que eso. Más aún, debido a su tamaño existe un costo mucho mayor en garantizar el mismo nivel de disponibilidad.
- Los data-centers típicamente corren muchos procesos controlados, usualmente llamados ETL (Extraction, Transformation and Loading por su sigla en inglés). Como resultado, las actualizaciones en los data-centers son mas conocidas y pueden ser reproducibles de las fuentes de datos.
- Un data-center típicamente guarda una gran cantidad de datos históricos, que usualmente no son objeto de cambio. Los datos que no cambian solamente necesitan ser respaldados una única vez.

Es necesario tener una estrategia de respaldos como parte del diseño del sistema y considerar que respaldar, y cada cuanto tiempo. Las variables mas portantes en el diseño de los respaldos son la cantidad de recursos existentes en la plataforma a la hora de realizar un respaldo o restauración, y el tiempo objetivo de la restauración (la cantidad de tiempo que es posible mantener al sistema inaccesible).

Las operaciones que no emiten logs deben ser tenidas en cuenta cuando se plantee una estrategia de respaldo y recuperación. En la recuperación tradicional, la

restauración de un respaldo y la aplicación de cambios de un log archivado no aplican para este tipo de operaciones. Teniendo en cuenta eso, las subsecuentes operaciones que impactan en los datos que fueron manipulados por la operación sin log fallan. Por lo tanto, este tipo de operaciones deben ser tomadas en cuenta cuando se diseña una estrategia de respaldo y recuperación.

Nunca se debe realizar un backup cuando esta siendo llevada a cabo una operación que no emite logs.

Es necesario planificar una de las siguientes estrategias, o una combinación de ambas:

- La estrategia ETL. Recuperar un backup que no contiene transacciones no-recuperables y repetir el ETL que ha tenido lugar entre el backup y la falla.
- La estrategia de backups incrementales. Realizar un backup inmediatamente después de que ha sucedido alguna transacción no recuperable. Oracle por ejemplo, provee de un servicio de tracking de archivos que habilita a hacer backups incrementales basados en los cambios de los bloques de datos. RMAN aprovecha el servicio de tracking de archivos.

La elaboración de una estrategia de respaldo y recuperación puede ser una tarea desalentadora, y cuando se tratan de cientos de gigabytes de datos que deben ser protegidos y recuperados en casos de falla, la estrategia puede ser muy compleja.

Estas son algunas de las mejores prácticas que pueden ayudar a implementar una estrategia de respaldo y recuperación en un data-center:

- Usar el modo Archivelog. Registros de logs para rearmar datos son cruciales para la recuperación cuando no se permite que los datos se pierdan, ya que constituyen un registro de cambios en la base de datos.
- Usar RMAN. Hay muchas razones para implementarlo, algunas son: grandes reportes, backups incrementales, backups sin bajar el servicio, valuación y optimización de los backups y restore, integración fácil con administradores de contenido, recuperación de bloques, validación y administración de logs archivados, y detección de bloques corruptos.
- Usar tablas de solo lectura. Uno de los principales problemas de los backups en los data-centers es el tiempo que puede llevar realizar un backup dado el tamaño de los mismos, los cuales pueden llevar horas. Por lo tanto, una consideración importante es minimizar la cantidad de datos a ser respaldados. Las tablas de solo lectura son el mecanismo mas simple para ello, dado que esos datos solo deben ser respaldados una vez. La mayoría de los data-centers mantienen datos en sus tablas que son particionados en rangos de tiempo. La información en general está “activa” por un periodo de tiempo determinado, desde 30 días a un año, todo lo anterior es de solo lectura, por lo

que basta con ser respaldado una vez.

- Plan para operaciones sin log. En general, una de las prioridades mas importantes de un data-center es la performance. Mas allá de la eficiencia con los usuarios, debe ser eficiente durante el proceso ETL (Extraccion, transformación y carga). Una optimización lograda es ejecutar operaciones en modo “NOLOGGING” (que no emiten logs). El hecho de no hacer log de seguridad hace que aumente la performance hasta en un 50%. Sin embargo, estas operaciones no son recuperables por los métodos tradicionales, dado que serian necesarios dichos logs. Por lo tanto, la presencia de operaciones sin log debe ser tomada en cuenta (en caso de ser utilizada).
- Tener en cuenta que no todas las tablas son igual de importantes. Mientras que el camino mas fácil e tomar a todas las tablas por igual, hay ciertas estrategias que toman la importancia de las mismas para aumentar la eficiencia. La granularidad básica de un respaldo y restauración es la tabla, por lo que distintas tablas podrían tener distintas estrategias. En el nivel mas básico, las tablas temporales nunca deberían ser respaldadas (una regla reforzada en RMAN). A otro nivel, quizás un tabla de ventas tenga mas importancia que la del flujo de clicks en el sistema por ejemplo; dado esto se podría implementar una estrategia que tenga en cuenta ese aspecto.

## Respaldando un data-center [11]

Un sistema de procesamiento de transacciones en línea (Online Transaction Processing - OLTP) captura los datos entrantes y actualiza una base de datos. Para asegurarse contra la pérdida de datos, el sistema registra las transacciones a medida que se van produciendo, y los administradores desarrollan estrategias de respaldo que incluyen backups completos cada cierto tiempo, e incrementales cada menos tiempo que los anteriores. Estas estrategias son diseñadas para prevenir la perdida de datos, minimizar la interferencia con los procesamientos de operaciones transaccionales y para proveer de una recuperación rápida en casos de fallas.

En contraste, un data-center guarda una cantidad muy grande de datos históricos estables que son actualizados en un horario y cada cierto periodo de tiempo determinado. Para los data-centers, es necesario diseñar estrategias de respaldo que minimicen los backups completos y que utilicen los backups incrementales para las actualizaciones de datos.

Las limitaciones de tiempo de recuperación suelen ser mas flexibles y menos restrictivas para las fallas en data-centers que para fallas en OLTP. Las limitaciones de tiempo mas permisivas para recuperar datos usualmente permiten a los data-centers realizar full-backups mucho menos frecuentemente que para los sistemas OLTP. Por ejemplo, una tabla de facturas puede contener cientos de millones de lineas que representan las ventas de 10 años. Es muy poco probable que

se realicen cambios en los datos de ventas después de que el negocio ha realizado un proceso de cierre de fin de año.

Hacer repetidamente backups de datos que no han cambiado resulta innecesario, y las estrategias de backup deberían tener esto en cuenta. Dependiendo de las limitaciones de tiempo y el volumen de los datos, debe ser creada una estrategia que respalde los datos agregados durante las actualizaciones del almacén de datos, usando backups incrementales, y entonces crear un backup solamente de los datos agregados durante el año actual después del cierre del año. Para recuperarse de una falla completa de la base de datos del data-center sería necesario cargar múltiples backups, uno por cada año anterior al año actual, y entonces realizar los backups incrementales para las actualizaciones del año actual.

“SQL Server 2000 Analysis Services” mantiene datos OLAP en las bases de datos del servidor de análisis con fines especiales, que pueden ser archivados y restaurados por separado de las copias de seguridad de bases de datos de almacenamiento de datos .

### **Backup en Facebook:** [12]

Facebook posee una de las instalaciones de MySQL más grandes del mundo, con miles de servidores de bases de datos en múltiples regiones. No es sorpresa que sea todo un reto realizar un respaldo. El objetivo buscado es el de guardar toda la información agregada por los usuarios de manera segura, asegurando que nada sea borrado, y todo lo borrado sea oportunamente eliminado de forma precisa.

Para conseguir esto, Facebook tiene construido un sistema automatizado y altamente efectivo que mueve varios petabytes (1000 terabytes) por semana. Sin hacer tanto foco en el testing, el sistema se centra en detectar fallas de manera rápida, y corregir de forma autónoma.

#### **Etapa 1: Los logs binarios y mysqldump**

La primera línea de protección se denomina "etapa 1" o copias de seguridad "en rack" (RBU). En cada rack de bases de datos, sin importar el tipo, hay dos servidores de almacenamiento RBU. Tener RBUs en el mismo bastidor que los servidores de bases de datos permite el utilizar el máximo ancho de banda y latencia mínima, y también actúa como un amortiguador para la próxima etapa de las copias de seguridad (más sobre esto más adelante).

Uno de los trabajos de estos servidores es recoger lo que se conoce como logs binarios. Estos logs son un registros de las actualizaciones de la base de datos segundo a segundo. Los logs binarios se transmiten constantemente a los anfitriones RBU por medio de un proceso esclavo simulado. Efectivamente, la RBU recibe las mismas actualizaciones que una réplica sin ejecutar mysql.

Mantenerse al día con los logs binarios en la RBU es importante: si un servidor de base de datos maestra (master) llegara a quedar desconectado, los usuarios de este servidor no podrían realizar actualizaciones de estado, subir fotos, etc. Estas fallas ocurren, pero el tiempo de reparación debe ser lo más corto posible para mantener a Facebook utilizable. Tener los logs binarios disponibles permite promover otra base de datos para convertirse en maestra en cuestión de segundos. Dado que se puede tener logs binarios en segundos disponibles en la RBU, se puede hacer esto aunque el viejo maestro no esté disponible. Tener logs binarios también permite la recuperación un punto en el tiempo mediante la aplicación de las transacciones registradas a una copia de seguridad anterior.

El segundo trabajo del servidor es el de realizar backups tradicionales. Existen dos formas de realizar un respaldo en MySQL server: binario y logico (mysqldump). Facebook utiliza el backup lógico, ya que son respaldos sin versionar, proveen mejor integridad de información, son mas compactos y toma mucho menos trabajo la restauración.

Una de las mayores ventajas de mysqldump es que la información en disco corrupta no es propagada al backup. Si un sector del disco se corrompe, o es escrito incorrectamente, el checksum de la InnoDB será incorrecto. Cuando se ensambla el backup, MySQL va a leer o bien el bloque correcto de la memoria, o ira al disco y encontrará el checksum mal, alertando el backup (y el proceso de la base de datos). Una desventaja de mysqldump es la contaminación de la cache LRU (least recently used), utilizada para bloquear el cacheo de InnoDB. Afortunadamente, versiones posteriores de MySQL incluyen código para mover las inserciones en la LRU de escaneos al final de la cache.

Cada RBU toma un backup de todas las bases de datos bajo su competencia. Mas alla del tamaño del ambiente, es posible completar un backup de todos los datos en unas pocas horas.

Si un RBU falla, un software automáticamente redistribuye su responsabilidad a otros sistemas en el mismo cluser. Cuando vuelve a estar online, esta responsabilidad retorna al RBU en el host original. En facebook no se preocupan demasiado por la retención de datos en sistemas individuales por la arquitectura de la etapa 2.

## **Etapa 2: Hadoop DFS**

Luego de cada respaldo y la colección de logs binario, inmediatamente se copia a

uno de los clusters Hadoop. Estos son clusters almacenes de datos replicados, altamente estables con tiempos de retención establecidos. Debido a que los tamaños de los discos crecen muy rápido, las generaciones antiguas de los RBU's pueden ser demasiado pequeñas para almacenar mas de un día o dos de backups, pero los clusters pueden ser tan grandes como sea necesario. La naturaleza distribuida de Hadoop provee el ancho de banda necesario para recuperar la información rápidamente.

Pronto Facebook comenzará a cambiar el análisis diferido de información a estos clusteres Hadoop. Esto reducirá el numero de lecturas no criticas en los servidores de bases de datos, dejando mas lugar para hacer a Facebook tan rápido como sea posible.

### **Etapa 3: Almacenamiento a largo plazo**

Una vez por semana, se copian los backups desde el Hadoop a almacenamiento secundario en otra región: esto es conocido como la etapa 3. Estos sistemas de almacenamiento son modernos y seguros, y quedan por fuera del flujo normal de trabajo.

### **Monitoreo**

Mas allá del sistema habitual de monitoreo, Facebook saca estadísticas específicas, como la latencia del log binario, la capacidad del sistema, etc.

Una de las herramientas más valiosas que tiene Facebook es guardar la severidad de una falla en los backups. Como es normal perder algún backup debido al tamaño del ambiente y la cantidad de mantenimiento en marcha (por ej: reconstrucción de replicas), se necesita una forma de encontrar fallas extendidas y backups individuales que no han tenido éxito en varios días. Por este motivo, la cantidad de backups perdidos se incrementa exponencialmente con el tiempo, y varias agrupaciones de estas cantidades nos dan un indicador útil y rápido del estado de la copia de seguridad.

Por ejemplo, una base de datos faltante por un día es un punto. 50 bases faltantes por día harían 50 puntos. Sin embargo, una base de datos faltante por 3 días serían 27 ( $3^3$ ) y 50 bases faltantes por 3 días, resultaría en un verdadero problema y requeriría de atención inmediata.

### **Restauración**

Para poder tener backups confiables estos deben estar testeados, si no es como no tenerlos. Por esto Facebook construye su sistema de forma que el mismo está realizando continuamente restauraciones desde la etapa 2 a el servidor. Despues de hacer una restauración, se corren varios controles de corrección sobre los datos, si hay algun problema repetido, se dispara una alarma para que haya una revisión humana. Este sistema puede detectar todo, desde bugs en MySQL hasta fallas en el proceso de backup, permitiendo ser mas flexible con lo que cambie en el ambiente

de backup.

Facebook también ha desarrollado un sistema llamado ORC (un coordinador de restauraciones recursivo y asíncrono) que permite un auto-servicio de restauración para ingenieros que quieran recuperar versiones anteriores de sus herramientas de bases de datos. Esto es especialmente útil para el desarrollo ágil.

## Sitios de alta performance.

Las redes sociales son aplicaciones web, y como tales necesitan ser rápidas. Deben de interactuar con el usuario de la forma más dinámica y responder a las acciones de los usuario lo mas rápido posible. Facebook, Google y Microsoft han concluido que los usuarios visitan más paginas y le dan mayor valor a los sitios cuando estos se ejecutan rápido. [13].

Entonces, como hacemos para determinar que tan rápido es un sitio? Podemos utilizar tres componentes principales que contribuyen a la carga de una pagina:

- Tiempo de red: representa el tiempo que un usuario esta esperando mientras los datos son transmitidos entre su computadora y el sitio. No siempre se puede controlar el tiempo de red ya que depende de la velocidad de las conexiones, pero se puede reducir el numero de bytes requeridos para cargar la pagina. Cuanto menos bytes menor el tiempo de red. Html, CSS, JavaScript, images y cookies son los 5 contribuyentes principales al tiempo de red.
- Tiempo de generación: es el tiempo que toma desde que el servidor web recibe el pedido del usuario hasta que envía la respuesta. Esto mide la eficiencia del código del servidor, el cacheo, la base de datos y el hardware de red. Este tiempo puede ser reducido mejorando el código, haciendolo mas sencillo, limpio, veloz , también mejorando la arquitectura del backend.
- Tiempo de renderizado: mide cuento tiempo necesita el navegador web para procesar la respuesta del servidor web y mostrar la pagina en pantalla. Es limitado por la performance y comportamiento de los navegadores pero se puede mejorar, al igual que para el tiempo de red, disminuyendo la cantidad de código enviado, minimizando los bytes de HTML, CSS, Javascript e imágenes. Otra manera de reducirlo es ejecutando lo menos posible de JavaScript antes de mostrar la pagina al usuario.

Cuanto más compleja se convierte un sitio Web, se vuelve más importante contar con

servidores y clientes que ofrecen un mejor rendimiento.

Facebook realizo varias mejoras para poder disminuir estas métricas a la mitad.

Primero que nada invirtieron mucho tiempo aplicando las buenas prácticas establecidas por los pioneros en el rendimiento de la web. La idea clave detrás de estas buenas prácticas es darse cuenta que sólo el 10-20% del tiempo total de respuesta al usuario final se gasta obteniendo el HTML en el navegador. Es necesario centrarse en el otro 80-90%, si se quiere hacer las páginas mucho más rápido. Estas reglas son algunas de las mejores prácticas para optimizar la forma en que los servidores y navegadores manejan ese 80-90% de la experiencia del usuario [14]:

1. Hacer la menor cantidad de pedidos HTTP
2. Utilizar una red de distribución de contenidos.
3. Agregar cabecal Expire a los componentes con fecha a futuro haciéndolos cacheables.
4. Comprimir los componentes con Gzip.
5. Poner los Stylesheets al comienzo
6. Poner los Scripts al final,
7. Evitar expresiones css
8. Poner el código css y Javascript en archivos externos.
9. Reducir las búsquedas de DNS mediante el uso de Keep-Alive y menos dominios.
10. Minificar el código Javascript.
11. Encontrar maneras de evitar las redirecciones.
12. Asegurarse que los script estén solo una vez.
13. Reconfigurar o remover ETags.[15]
14. Hacer Ajax Cacheable.

Facebook se enfoco en 2 objetivos [16]: reducir drásticamente la cantidad de bytes de cookies, HTML, CSS y JavaScript como también desarrollar nuevos frameworks y metodologías que permitan al navegador mostrar el contenido de la pagina lo más rápido posible.

Para reducir HTML y CSS los ingenieros de Facebook desarrollaron una nueva librería de componentes reutilizables . Antes del desarrollo de esta biblioteca de componentes, cada pagina se basaba en una gran cantidad de HTML y CSS personalizado, aunque muchas paginas compartían características y funcionalidades similares. Con esta librería es mas sencillo optimizar el HTML en un solo lugar y luego verse reflejado en todo el sitio. Otro beneficio es que, los componentes comparten reglas CSS, una vez que un usuario ha descargado un poco de CSS es muy probable que esas reglas se vuelvan a utilizar en la pagina siguiente sin tener la necesidad de volver a descargarlas. Con estos cambios, bajaron el promedio de bytes CSS por pagina un 19% y un %44 de bytes HTML por página. Con estos cambios significativos se logra que los usuarios obtengan el contenido mas velozmente y los navegadores lo procesen mas rápido.

Facebook se siente dinámico y fluido gracias a las funcionalidades creadas en

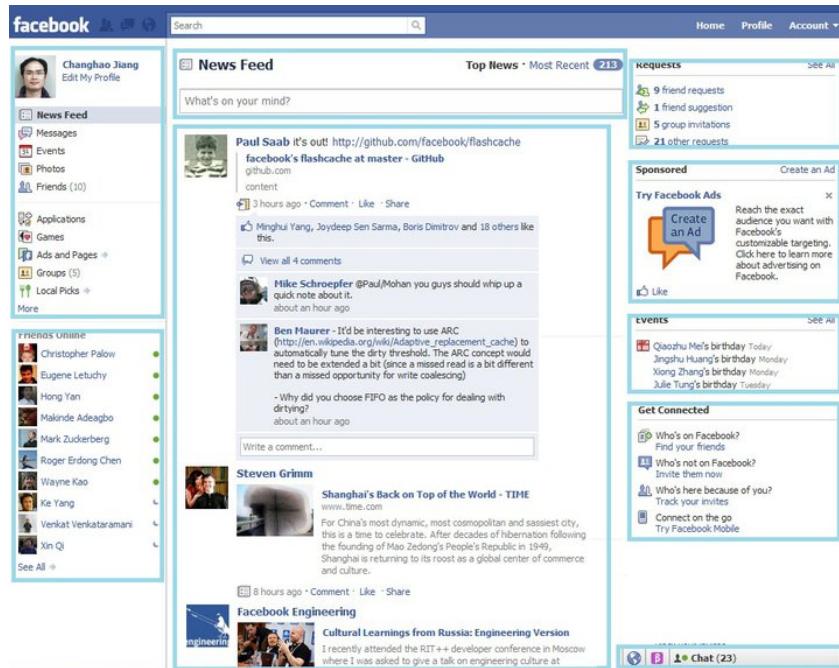
JavaScript. Pero a medida que se agregaban mas y mas características se debió escribir mas Javascript el cual debía ser descargado por los usuarios para ser utilizado. Agruparon un conjunto de funcionalidades que eran utilizadas en la mayoría de las funciones en una eficiente y pequeña librería cacheable llamada Primer. Reescribieron la mayoría del código Javascript en torno a esa librería logrando disminuir la cantidad de bytes de JavaScript en un 40% por pagina.

## **BigPipe**

El modo tradicional de cargar una pagina incluye enviar un pedido al servidor, el servidor genera la respuesta que envía al navegador y el navegador convierte la respuesta en algo que se puede ver e interactuar. Analizando este modelo, mientras el servidor prepara y envía de vuelta la respuesta al navegador, este se encuentra ocioso, esperando algo para hacer. Una buena idea seria, que el servidor enviara una respuesta parcial al navegador, el cual podría empezar a descargar JavaScript y CSS y comenzar a mostrar un poco de contenido. Una vez que el servidor haya procesado un poco más de la respuesta, se le envía de vuelta al navegador y así hasta que el servidor termina el trabajo procesando toda la respuesta. De esta manera sobreponemos una porción significante del tiempo de generación con el tiempo de renderizado, disminuyendo el tiempo de espera del usuario.

Facebook ha implementado esta capacidad, desarrollando un nuevo formato de transferencia y el procesamiento en paralelo entre el servidor y el cliente. A esta solución se le llamó Big Pipe.

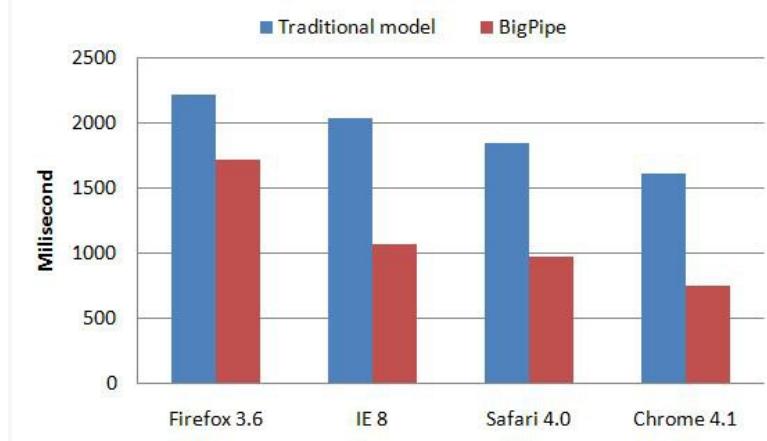
Big Pipe es el sistema completo que permite dividir las paginas en bloques lógicos de contenido, llamado Pagelets, y permite la generación y renderizado de estos Pagelets en paralelo. A continuación se puede ver como una pagina de Facebook es dividida en pequeños Pagelets.



En la figura las cajas celestes representan los Pagelets. Cada uno de ellos tienen un ciclo de vida independiente. Mientras que el menú en la parte izquierda de la página se muestra al usuario, el pagelet "News Feed" podria estar siendo generada en el servidor.

BigPipe no solo reduce el tiempo completo de carga de una pagina, sino que también lo hace parecer aún más rápido para los usuarios ya que se va renderizando la pagina parcialmente a medida que va siendo procesada y no es necesario que se despliegue todo una vez finalizada la descarga.

El siguiente gráfico muestra los datos de rendimiento que comparan la latencia percibida por el usuario en ver el contenido total de la página de inicio de Facebook, una en el modelo tradicional y otra con BigPipe. El gráfico muestra que en la mayoría de los navegadores, BigPipe reduce la latencia percibida por el usuario a la mitad.



## **Big Data**

Debido a la masividad y el tráfico que adquieren las redes sociales se generan grandes volúmenes de datos. El análisis de los mismos deja de ser trivial como consecuencia de su gran tamaño. Para ello se han desarrollado una serie de herramientas que permiten analizar esos grandes volúmenes de datos en tiempos de procesamiento aceptables.

### **¿Qué es big data?**

Según IBM:

– “...enormes cantidades de datos (estructurados, no estructurados y semi estructurados) que tomaría demasiado tiempo y sería muy costoso cargarlos en una base de datos relacional para su análisis”.

Según Wikipedia:

– “Es el sector de las tecnologías de la información que referencia a sistemas que manipulan grandes volúmenes de datos (o data sets).”

Uno de los conceptos básicos para trabajar con grandes volúmenes de datos se llama map reduce. Se basa a grandes rasgos en dividir los datos en porciones (map) para que puedan ser procesados por unidades de cómputos (nodos), generando cada una un resultado intermedio que se unificará (reduce) con el resto de los resultados intermedios para obtener el resultado final .

A modo de ejemplo Facebook utiliza herramientas de Apache como ser Hadoop, Hbase, Hive (migrado a Presto), Scribe [17]

En cuanto a la proyección a futuro podemos agregar que Según un informe de Gartner, el “Big Data” creará 4.4 millones de puestos de trabajo para el 2015, pero habrá gente solo para cubrir 1/3 de ellos. [18]

## **Hadoop**

La biblioteca de software Apache Hadoop es un framework que permite el procesamiento distribuido de grandes conjuntos de datos a través de grupos de ordenadores que utilizan modelos de programación simples. Está diseñado para pasar de los servidores individuales a miles de máquinas, cada uno con la computación y el almacenamiento local. En lugar de confiar en el hardware para ofrecer alta disponibilidad, la biblioteca en sí está diseñada para detectar y controlar los errores en la capa de aplicación, por lo que la entrega de un servicio de alta disponibilidad en la parte superior de un grupo de computadoras, cada una de las cuales pueden ser propensos a fallas.[19]

Esta compuesto por Hadoop Common que incluye las librerías y utilitarios para dar soporte a el resto de los módulos Hadoop. HDFS (Hadoop Distributed File System) sistema de archivos distribuido que provee un acceso de alto rendimiento a los datos. Hadoop Yarn un framework para la planificación de tareas y gestión de recursos de clúster. Hadoop MapReduce Sistema basado en Yarn para el procesamiento en paralelo de grandes conjuntos de datos.

[20]

### **Los cinco demonios de Hadoop**

1. NameNode: guarda los metadatos para el HDFS
2. NameNode secundario:  
Realiza las funciones de limpieza para el NameNode  
No es un backup del NameNodo principal
3. DataNode: Almacena los Bloques de Datos HDFS actuales
4. JobTracker: gestiona los Jobs de MapReduce y distribuye las tareas individuales.
5. TaskTracker: es responsable de instanciar y monitorizar los Map individuales y las tareas Reduce

[26]

## **HBase**

Apache HBase™ es la base de datos Hadoop, distribuida y escalable . Se utiliza cuando se necesita, tiempo real acceso de lectura / escritura aleatoria a Big Data. La meta de este proyecto es la organización de tablas muy grandes. Así como Bigtable aprovecha el almacenamiento de datos distribuidos proporcionada por el Google File System, Apache HBase proporciona capacidades Bigtable sobre Hadoop y HDFS. HBase nos permitirán tener los datos distribuidos a través de lo que denominan regiones. Una región no es más que una partición tipo Nodo de Hadoop

que se guarda en un servidor. La región aleatoria en la que se guardan los datos de una tabla se decide por nosotros, dándole un tamaño fijo a partir del cual la tabla debe distribuirse a través de las regiones.[26]

## HDFS

HDFS es un sistema de ficheros distribuido, escalable y portátil escrito en Java y creado especialmente para trabajar con ficheros de gran tamaño. Una de sus principales características es un tamaño de bloque muy superior al habitual (64 MB) para no perder tiempo en los accesos de lectura. Los ficheros que normalmente van a ser almacenados o ubicados en este tipo de sistema de ficheros siguen el patrón “Write once read many” (escribe una vez y lee muchas), de este modo, está especialmente indicado para procesos batch de grandes ficheros de logs por ejemplo, los cuales sólo serán escritos una vez y, por el contrario, serán leídos gran cantidad de veces para poder analizar su contenido. Igualmente, se puede aplicar este concepto a bases de datos orientadas al almacenamiento masivo de datos como HIVE o Cassandra.

Por tanto, con HDFS tenemos un filesystem distribuido y especialmente optimizado para almacenar grandes cantidades de datos, siguiendo en todo momento con la filosofía de Apache Hadoop. De este modo, los ficheros serán divididos en bloques de un mismo tamaño y distribuidos entre los nodos que forman el clúster de datos (los bloques de un mismo fichero pueden caer en nodos distintos).

Un clúster HDFS tiene dos tipos de nodos, diferenciados completamente según el rol o la función que vayan a desempeñar a la hora de ser usados. Los dos tipos de nodos HDFS son los siguientes:

Namenode (JobTracker): Este tipo de nodo, del que sólo hay uno por clúster, es el más importante ya que es responsable de la topología de todos los demás nodos y, por consiguiente, de gestionar el espacio de nombres. En este nodo se almacenan los metadatos.

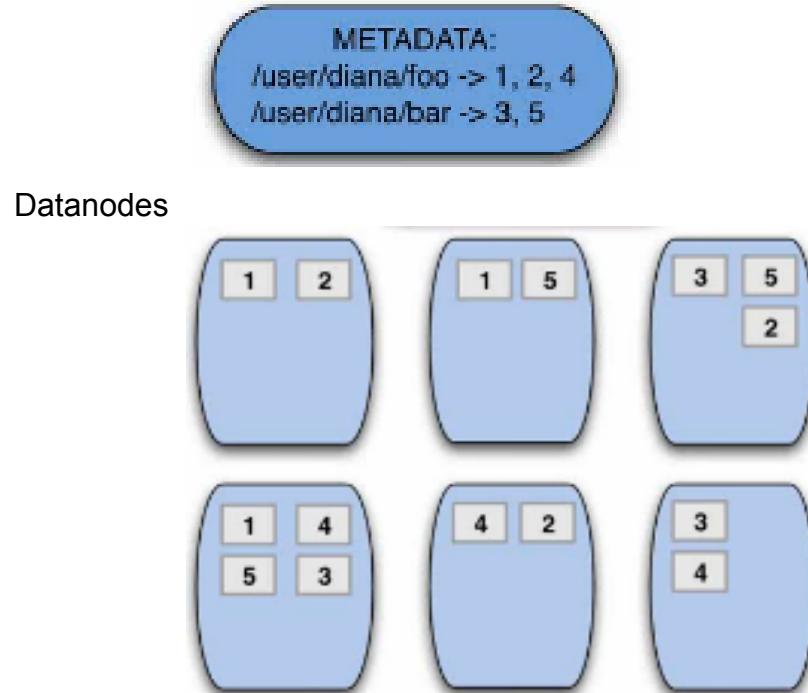
Datanodes (TaskTracker): Este tipo de nodos, de los que normalmente van a existir varios, son los que realizan el acceso a los datos propiamente dicho. En este caso, almacenan los bloques de información y los recuperan bajo demanda.

Simplificando, se puede considerar el JobTracker como el nodo principal o director de orquesta, mediante el cual se va a distribuir el tratamiento y procesado de los ficheros en los TaskTracker, o nodos worker, que realizarán el trabajo.

[21]

## Ejemplo

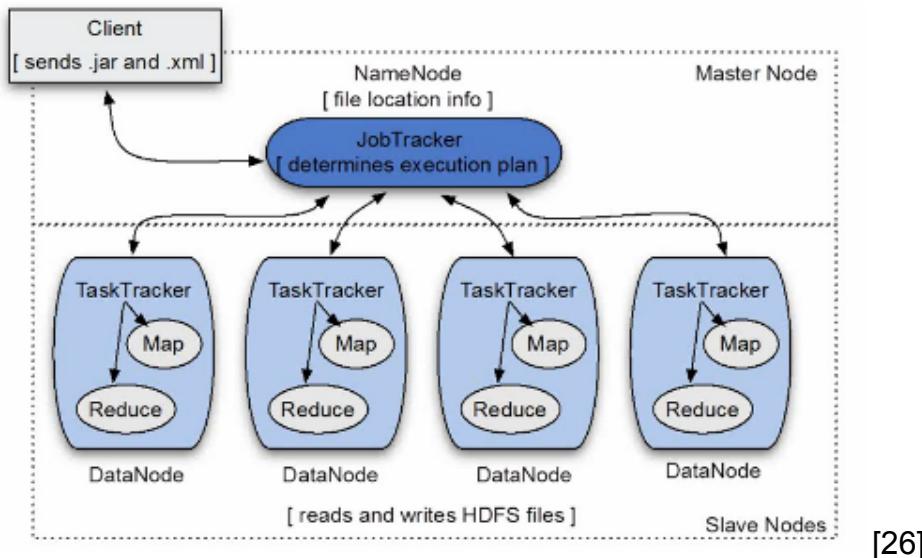
Namenode



[26]

De esta manera hadoop provee de un sistema de archivos orientado a grandes volúmenes de datos.

## Configuración Básica de un Cluster



## Yarn

YARN (Yet Another Resource Negotiator) es básicamente una versión mejorada de la arquitectura MapReduce. Es por esto que también se le suele denominar MapReduce 2.0 (MRv2) y está disponible desde la distribución Hadoop 0.23 en adelante.

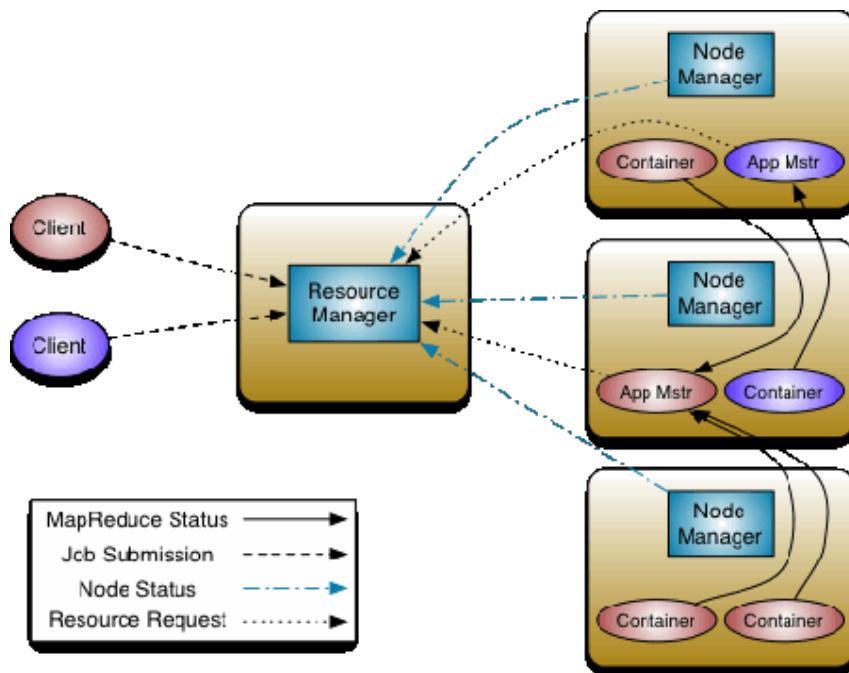
Se crea para dividir las dos funciones principales del JobTracker (NameNode), es decir, tener en servicios o demonios totalmente separados e independientes la gestión de recursos por un lado y, por otro, la planificación y monitorización de las tareas o ejecuciones.

Hadoop HDFS es la capa de almacenamiento de datos para Hadoop y MapReduce era la capa de procesamiento de datos. Sin embargo, el algoritmo de MapReduce, por sí solo, no es suficiente para la gran variedad de casos de uso que vemos. Hadoop se emplean para resolver. Con YARN, Hadoop ahora tiene un entorno de gestión de recursos y aplicaciones distribuidas donde se pueden implementar múltiples aplicaciones de procesamiento de datos totalmente personalizadas y específicas para realizar una tarea en cuestión.

De esta separación surgen dos elementos:

- **ResourceManager (RM):** Este elemento es **global** y se encarga de toda la gestión de los recursos.
- **ApplicationMaster (AM):** Este elemento es **por aplicación** y se encarga de la planificación y monitorización de las tareas.

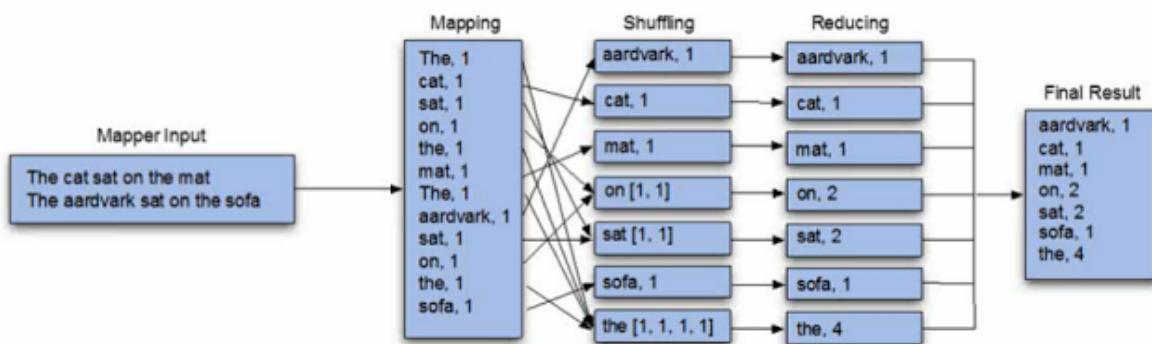
## Arquitectura de YARN



De este modo, el ResourceManager y el NodeManager (NM) esclavo de cada nodo forman el entorno de trabajo, encargándose el ResourceManager de repartir y gestionar los recursos entre todas las aplicaciones del sistema mientras que el ApplicationMaster se encarga de la negociación de recursos con el ResourceManager y los NodeManager para poder ejecutar y controlar las tareas, es decir, les solicita recursos para poder trabajar.

## Fase de MapReduce

Ejemplo



## Hive

Apache Hive es un framework con soporte para data warehouse sobre hadoop el cual provee summarización de datos , consultas y análisis de los mismos almacenados en sistemas de archivos compatibles Permite agregarle estructura a la información presente en Hadoop y consultar esta información en un lenguaje muy similar a SQL , permite además reutilizar mapers y reducers pre existentes. Se pueden ejecutar tareas MapReduce sin escribir código, únicamente realizando una consulta HiveQL.

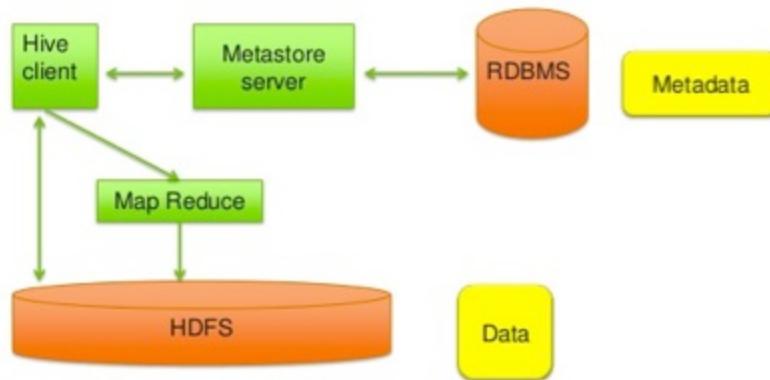
Ejemplo - SELECT a.foo FROM invites a WHERE a.ds='2008-08-15';

Incialmente fue desarrollado por Facebook y actualmente es utilizado por otras compañías como Netflix.[22] [23]

Amazon mantiene una rama de Apache Hive la cual se encuentra incluida en Amazon Elastic MapReduce sobre Amazon Web Services.

Facebook sustituyó Hive por Presto , proyecto el cual abrió (open source)[24]

## Arquitectura



GUI para Apache Hadoop [26]

- Permite realizar consultas HiveQL, probar ejecución de consultas, ver jobs ejecutando, mantener histórico de consultas, ver la generación de los mappers por Hive y la ejecución de los mismos, entre otros.
- Permite además ver histórico de ejecuciones y resultado de las mismas gráficamente.
- Fácilmente se puede verificar si una nueva consulta de Hive es correcta sin código.

The screenshot shows the Hue Query Editor interface. The title bar says "Hue - Query". The main area is titled "Query Editor : Job loss (copy)" with the subtitle "Job loss among the top earners 2007-08". The query code is as follows:

```

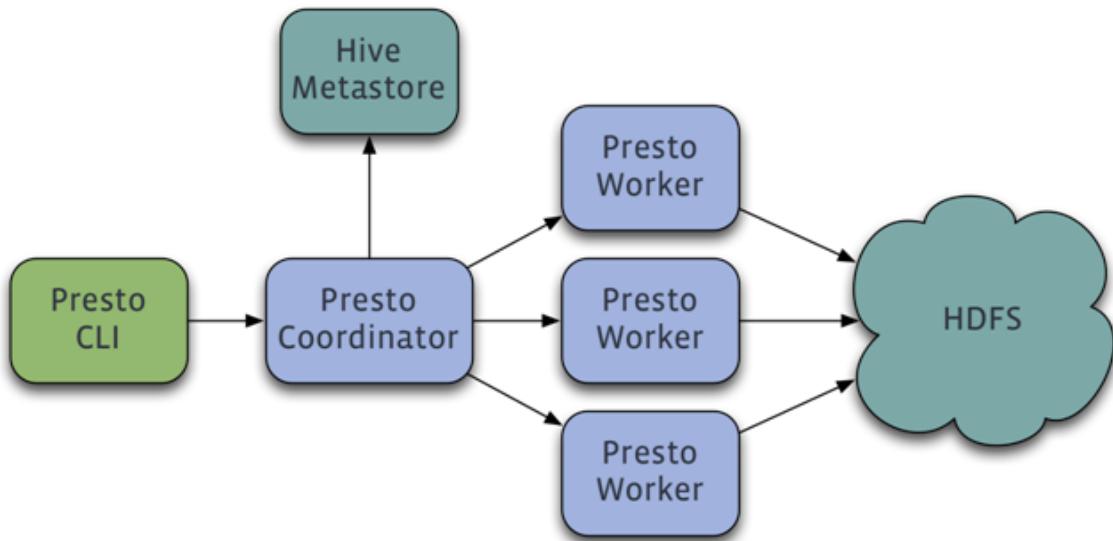
1 SELECT s07.description, s07.total_emp, s08.total_emp, s07.salary
2 FROM
3 sample_07 s07 JOIN
4 sample_08 s08
5 ON 1 AND s07.code = s08.code
6 WHERE
7 1 AND s07.total_emp > s08.total_emp
8 AND s07.salary > 100000
9 ORDER BY s07.salary DESC

```

On the left side, there are sections for "DATABASE" (set to "default"), "SETTINGS" (with "Add" button), "FILE RESOURCES" (with "Add" button), "USER-DEFINED FUNCTIONS" (with "Add" button), "PARAMETERIZATION" (checkbox "Enable Parameterization" is unchecked), and "EMAIL NOTIFICATION" (checkbox "Email me on completion" is unchecked). At the bottom, there are buttons for "Execute", "Save", "Save as...", "Explain", "or create a", and "New query".

## Presto

Presto es un sistema distribuido que se ejecuta en un cluster de computadores. La instalación completa incluye un coordinador y multiples workers (instancias de ejecución que procesan datos). La consultas son realizadas desde un cliente (Presto CLI) hacia el coordinador (Presto Coordinator). El coordinador parcea analiza y planifica la ejecución de la consulta, luego distribuye el procesamiento a los workers.



## Requerimientos

- Linux or Mac OS X

- Java 7, 64-bit
- Python 2.4+

### Conectores

Presto soporta conectores pluginizables que proveen datos para consultar. Los requerimientos de los mismos varian según el conector

### Hadoop / Hive

Presto soporta lectura sobre datos de Hive a partir de las siguientes versiones de Hadoop:

- Apache Hadoop 1.x
- Apache Hadoop 2.x
- Cloudera CDH 4
- Cloudera CDH 5

### Soporta los siguientes formatos de archivos

- Texto
- Archivo secuencial
- RCFile
- ORC

Además el metastore de Hive es requerido. Los modos Local o embbebido no son soportados. Presto no utiliza MapReduce y por lo tanto sólo requiere HDFS.

### Cassandra

Cassandra 2.x es requerido. Este conector es completamente independiente del conector de Hive y solamente requiere de la existencia de la instalación de Cassandra.

### TPC-H

TPC-H connector genera datos dinámicamente que pueden ser utilizados para experimentar y testear Presto. El mismo no tiene requerimientos externos.

### Ejemplos de Consultas

Example:

```
SELECT *
FROM dim_all_users
TABLESAMPLE BERNoulli (50);
```

```
SELECT *
FROM dim_all_users
```

```

TABLESAMPLE SYSTEM (75);
Using with joins:
SELECT t1.*, t2.*
FROM
table1 t1 TABLESAMPLE SYSTEM (10)
JOIN
table2 t2 TABLESAMPLE BERNOULLI (40)
ON
t1.id = t2.id

```

## Salidas a Pantalla

```

presto:default> select n_nationkey, n_name from nation limit 5;
n_nationkey | n_name
-----+-----
0 | ALGERIA
1 | ARGENTINA
2 | BRAZIL
3 | CANADA
4 | EGYPT
(5 rows)

```

Query 20131105\_005533\_00081\_ee7y3, FINISHED, 2 nodes  
Splits: 2 total, 2 done (100.00%)  
0:00 [30 rows, 2.29KB] [88 rows/s, 6.76KB/s]

```

presto:default> select n_name, count(*) customers
-> from customer
-> join nation on (c_nationkey = n_nationkey)
-> group by 1
-> order by 2 desc
-> limit 5;
n_name | customers
-----+-----
VIETNAM | 6003717
MOROCCO | 6003115
ROMANIA | 6002183
CHINA | 6001991
IRAN | 6001889
(5 rows)

```

Query 20131105\_005539\_00082\_ee7y3, FINISHED, 13 nodes  
Splits: 187 total, 187 done (100.00%)

## **Migracion desde Hive**

Presto utiliza la sintaxis y semántica de ANSI SQL , mientras que Hive utiliza un lenguaje similar a SQL llamado HiveQL (que en sí tiene muchas diferencias con ANSI SQL).

Ejemplo de cambios para migrar

Consulta Hive:

```
SELECT a.id, b.userid  
FROM a  
LEFT JOIN b  
ON a.id = b.id AND a.ds = '2013-11-11'
```

Consulta Presto:

```
SELECT a.id, b.userid  
FROM a  
LEFT JOIN b  
ON a.id = b.id  
WHERE a.ds = '2013-11-11'
```

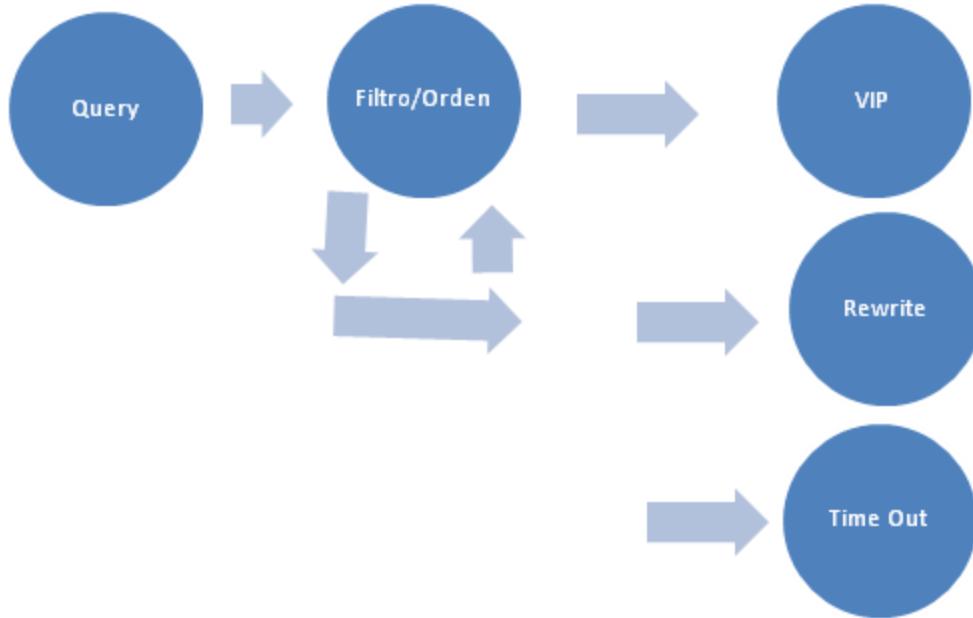
## Aplicaciones Reales Big Data

### Big data en Mercado Libre

- Hadoop: distribución de Cloudera.
  - Hay 85 nodos, c/u con 2x Hexacore => 1020 nucleos en total
  - Más de 3TB de RAM en total
  - Se genera +2TB/diarios
  - Hay +750TB en disco
  - Tecnologías utilizadas:
    - Hadoop | MapReduce | Hive | oozie | pig
    - Interfaz gráfica: Hue
- Dejavu: sistema de trackeo donde se guarda toda la información de visitas.
- ¿Qué guardamos en Dejavu?
    - Tipo de pagina, browser, búsqueda realizada, referer, user id, nickname, categoría del ítem, etc.

### Map Reduce aplicado a Busquedas

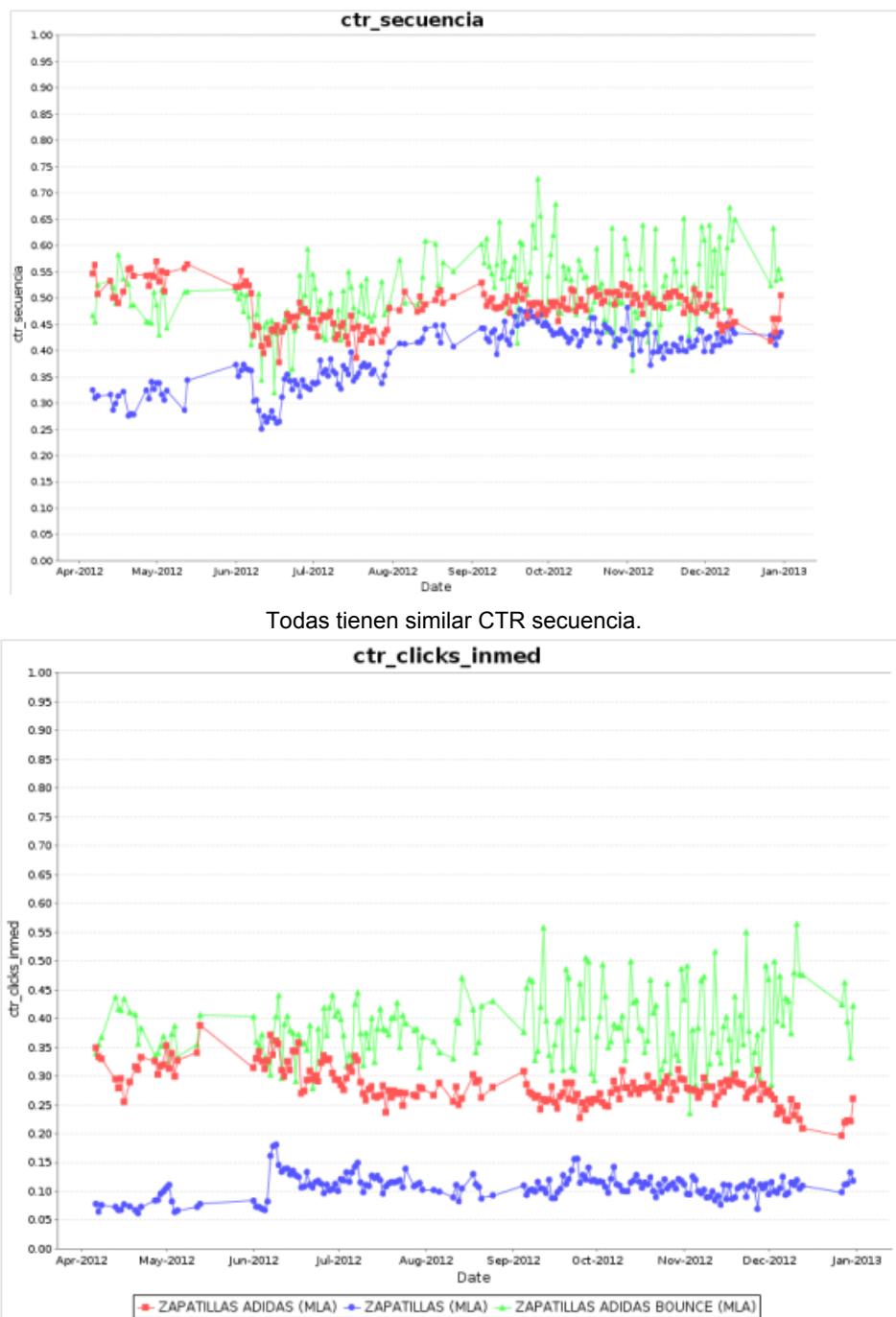
Flujo desde que se realiza una consulta , se consulta se aplican filtros u ordenan (una o mas veces) y ingresa en un link encontrado (VIP) reescribe la consulta o se lanza un time out.



CTR Secuencia: cant. secuencias VIP / cant. secuencias  
– “Cantidad de secuencias que terminan en una publicación”

• CTR Inmediato: Cant. Clicks inmediatos / cant. Secuencias

–“Clicks en un ítem inmediatamente luego de una búsqueda”



Consulta más específica de ZAPATILLAS ADIDAS BOUNCE tiene mucho mayor clicks inmediatos al ser una query más específica.

## Mejora de performance de las SERPs (Search engine results page)

Autos a control remoto eléctricos , autos a control remoto drift , autos a explosión usados , autos a control remoto usados ,

Artículos: 1-50 de 404547 | Más relevantes ▾

**Categorías**

- Juegos y Juguetes (12660)
- Autos y Camionetas (71475)
- Autos de Colección (1215)
- Coleccionables y Ho... (36255)
- Accesorios para A... (212435)
- Electrónica, Audio y ... (18900)
- Bebés (9125)
- Libros, Revistas y Co... (8435)
- Antigüedades (4975)
- Más opciones ▾

**Ubicación**

- Buenos Aires (208340)
- Capital Federal (169020)
- Santa Fe (8350)
- Córdoba (6915)
- Entre Ríos (3465)
- Mendoza (1635)
- Misiones (780)

**MercadoClics** | Llevá Tu Auto 0km Ahora - www.maynaronline.com - Maynar Ofrece Plan De Financiación Para Tu Nuevo 0km V... **MAYNAR** Anuncia aquí

**Renault Gordini 5p Da-3v 1969 Automotores El Puente** \$ 57.800<sup>00</sup> Artículo usado Buenos Aires

**Auto Mini Kart A Radio Control Remoto Rc Motor Electrico** \$ 399<sup>99</sup> 12 cuotas de \$ 49<sup>66</sup> Artículo nuevo 111 vendidos Capital Federal

**Volkswagen Vw Vento 2,0 Tdi 4p 2011 El Puente Automotore** \$ 158.000<sup>00</sup> Artículo usado Buenos Aires

Autos usados baratos , autos chocados , peugeot 206 , chevrolet corsa , bmw

Artículos: 1-48 de 71769 | Más relevantes ▾

**Inicio > Autos, Motos y Otros > Autos y Camionetas > "autos"**

**Marca**

- Volkswagen (12752)
- Renault (10227)
- Fiat (9529)
- Ford (8779)
- Chevrolet (7356)
- Peugeot (7339)
- Citroën (3284)
- Toyota (1551)
- Honda (1310)
- Más opciones ▾

**Modelo**

- Gol (1987)
- Ecosport (1692)
- Bora (1510)
- Ranger (1470)
- Clio (1434)
- Palio (1375)
- 206 (1331)
- Uno (1274)
- Gol Trend (1221)
- Más opciones ▾

**MercadoClics** | Buscás Una Camioneta? - www.ford.com.ar - Elegí Tu Ford Ranger Con Tasa 11,90 Y Disfrutá De Esta Financiación! **FORD** Anuncia aquí

Imagen	Título	Precio	Años
	Audi A4 1.8t Fsi (160cv) Multit...	\$ 168.500	2009   39.600 Km
	Vendo Ford Falcon 3.0 Con Gnc	\$ 10.000	1980   1.111 Km
	Volkswagen Bora 1.8 T Cuero /...	\$ 118.000	2009   53.000 Km
	Fiat Siena 1997 Se Vendio	\$ 27.500	1997   190.000 Km

Utilizando la información de los resultados inmediatos se selecciona la categoría correspondiente.

**mercado libre**

**zapatillas**

□ Solo en Zapatillas  Search icon

Regístrate Ingresar Vender Help icon

Búsquedas relacionadas: zapatillas dc , zapatillas nikes mujer , zapatillas reebok mujer , zapatillas supra , botines adidas

Inicio > Ropa y Accesorios > Zapatos > "zapatillas"

Artículos: 1-48 de 19148 □ | Más relevantes ▾

Género	MercadoClics   Queres Unas Zapatillas Dc - doshoes.com.ar - Compra On-Line Y Recibite Gratis Todos Los Medios De Pago, 12 Cuotas	Anuncia aquí
Hombre (9356)		
Mujer (3823)		
Niños (3894)		
Marca		
Adidas (2798)		
Converse (5078)		
Nike (616)		
Reebok (524)		
Topper (816)		
Puma (805)		
Vans (543)		
Asics (386)		
Stone (361)		
Más opciones ▾		
Estilo		
Urbanas (9620)	<b>Zapatilla De Running Topper L...</b> \$ 299 00 <span>Color</span> 12 cuotas de \$ 37 12	
Running (2871)	<b>Salomon  Cross Running J(trail...)</b> \$ 640 00 12 cuotas de \$ 79 40	
Botas (2096)	<b>Zapatillas Vans Old School Sk...</b> \$ 649 00 <span>Color</span> 12 cuotas de \$ 80 73	
Skate (1138)	<b>Zapatillas Adidas Porche Des...</b> \$ 699 00 12 cuotas de \$ 86 79	
Tenis (547)		
Trailing (467)		
Náuticas (208)		
Básquet (261)		
Otras (4782)		
Talle		
16 17 18 19 20		
21 22 23 24 25		
25.5 26 26.5 27 28		
Más opciones ▾		

MLA ▾ Query: ZAPATILLAS

Period: 20120606 - 20130606 (365 days), minimum clicks: 25

- Categories demand (745 users, 2 days actually considered)
  - Ropa y Accesorios (1430) **55.70%** (415 users)
  - Deportes y Fitness (1276) **39.06%** (291 users)

---

- Attributes demand (511 users)
  - 9993724 (9993724) **53.42%** (273 users)

## Miniconfig

- Ropa y Accesorios (1430) **91.0%**
- Zapatillas (109027) **91.0%**
  - Hombre (22389) **56.3%**
    - Adidas (6586) 12.5%
    - Mujer (22388) **28.7%**

**mercado libre**

zapatillas adidas

Búsquedas relacionadas: zapatillas adidas mujer , zapatillas puma , conjunto adidas , celulares usados , celulares baratos

Inicio > Ropa y Accesorios > Zapatillas > "zapatillas adidas"

Filtrado por: Adidas

MercadoClicks | Zapatillas Dc De Mujer - www.doshoes.com.ar - Envío Gratis A Todo El País Hasta 12 Cuotas Sin Interés | Anúnciate aquí

Artículos: 1-48 de 2790 | □ | Más relevantes ▾

Género	Categorías	Imagen	Título	Precio	Opciones
Hombre (1881)	Zapatillas Botas Bottas Adidas...		Zapatillas Botas Bottas Adidas...	\$ 749,99	12 cuotas de \$ 93,12
Mujer (688)	Bottas Adidas S Sleek Con Mo...		Bottas Adidas S Sleek Con Mo...	\$ 850,00	12 cuotas de \$ 106,50
Niños (291)	Zapatillas Adidas Porche Desi...		Zapatillas Adidas Porche Desi...	\$ 699,00	12 cuotas de \$ 80,75
Estilo	Zapatillas Adidas Skneo Grind...		Zapatillas Adidas Skneo Grind...	\$ 832,00	12 cuotas de \$ 103,99
Urbanas (347)					
Running (651)					
Botitas (417)					
Tenis (123)					
Básquet (70)					
Skate (41)					
Trailing (21)					
Náuticas (7)					
Otras (618)					
Talle	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30				
Más opciones ▾					
Color Primario					

□ **Attributes demand (832 users)**

- 999109027 (999109027) **58.05%** (483 users)
- 9993724 (9993724) **34.50%** (287 users)

MLA ▾ Query: ZAPATILLAS ADIDAS Submit

Period: 20120606 - 20130606 (365 days), minimum clicks: 25

□ **Categories demand (1025 users, 2 days actually considered)**

- Ropa y Accesorios (1430) **63.51%** (651 users)
- Deportes y Fitness (1276) **29.95%** (307 users)

□ **Autoselect**

- Category: 109027
- **Attributes:**
  - ADIDAS (999109027-AMLA\_109027\_2-MMLA33458)

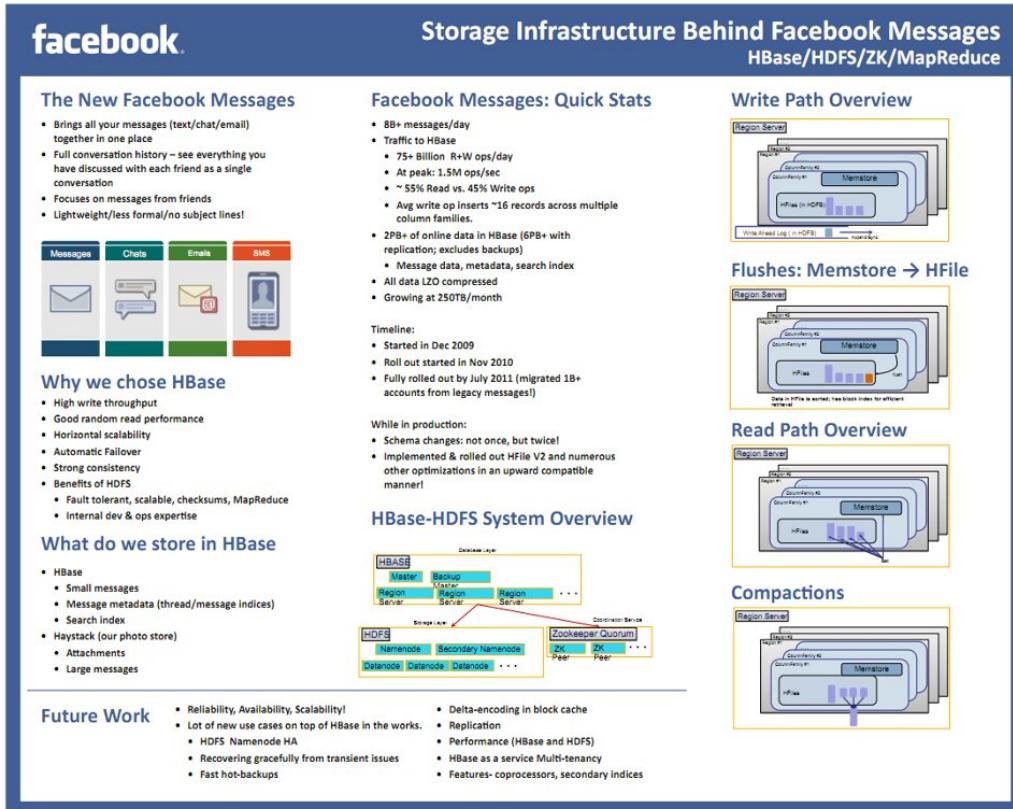
□ **Miniconfig**

- Ropa y Accesorios (1430) **92.1%**
  - Zapatillas (109027) **92.1%**
    - Hombre (22389) **50.8%**
    - Mujer (22388) **23.7%**

Y para calcular los crt a través del comportamiento de los usuarios es que se procesan los logs, y como estos en sitios de gran porte como las redes sociales son muy grandes es que se utiliza las soluciones de Big data presentadas.

## Big data en Facebook

### Big Data > Small Pieces



Otras aplicaciones Big Data se pueden ver en [25]

## 4. Conclusiones

Existen muchas tecnologías detrás del manejo de redes sociales. Esto se debe a que este tipo de sistemas tan grandes presentan muchos problemas a resolver, que no fueron previstos al momento de diseñarlos inicialmente, y que las tecnologías utilizadas para resolverlos no estaban pensadas para la magnitud de datos que terminaron manejando las redes sociales. Esto llevó a que diseñaran sus propias tecnologías y a extender las tecnologías ya existentes. Por ejemplo, en Facebook comenzaron utilizando memcached, pero como era escalable hasta cierto punto, diseñaron la tecnología TAO. Lo mismo sucede con las metodologías de backup.

Por otro lado, para poder enfrentar los nuevos retos que conllevan las redes sociales, se suelen dividir los problemas según las diferentes áreas arquitectónicas del sistema.

Se ha visto que las diferentes redes sociales estudiadas afrontan básicamente los mismos problemas, que incluso ha llevado a que grandes empresas (como Google, Facebook, LinkedIn, etc) se unan formando comunidades para desarrollar tecnologías para mutuo beneficio.

Otro punto a destacar es que cuando tienen que tomar una decisión entre disponibilidad y consistencia, las redes sociales generalmente priorizan la disponibilidad de sus sistemas, aunque en ciertos momentos estos no sean consistentes.

Ante el incremento exponencial de usuario y datos en las redes sociales, estas están en desarrollo constante, siempre buscando formas de mejorar lo que ya existe, y de crear soluciones tratando de anticiparse a los futuros problemas que puedan llegar a ocurrir. Incluso arriesgándose a utilizar como ambiente de testeo el entorno de producción.

## Referencias

[1] -

<http://www.merca20.com/las-6-mejores-redes-sociales-de-acuerdo-con-el-numero-de-usuarios-activos/>

Granja de Servidores

[2] - [http://technet.microsoft.com/en-us/library/cc526018\(v=office.12\).aspx](http://technet.microsoft.com/en-us/library/cc526018(v=office.12).aspx)

[3] - <http://www.slideshare.net/ottovaldez1/granja-de-servidores-web>

[4] -

<http://www.networkworld.es/social-business/el-tiempo-de-respuesta-de-las-redes-sociales-casi-se-triplica>

[5] - [http://content.time.com/time/photogallery/0,29307,2036928\\_2218548,00.html](http://content.time.com/time/photogallery/0,29307,2036928_2218548,00.html)

Tao

[6] - <http://elogia.net/blog/social-graph-facebook/>

[7] -

<http://www.cs.cmu.edu/~pavlo/courses/fall2013/static/papers/11730-atc13-bronson.pdf>

Mecanismos de distribucion de carga

[8] - [http://www.bdigital.unal.edu.co/3497/1/32296964.2009\\_1.pdf](http://www.bdigital.unal.edu.co/3497/1/32296964.2009_1.pdf)

[9] - <http://velocityconf.com/velocity2013/public/schedule/detail/28410>

Mecanismos de respaldo y recuperacion

[10] -

<http://serverfault.com/questions/203226/how-do-large-companies-backup-their-data>

[11] - [http://technet.microsoft.com/en-us/library/aa906012\(v=sql.80\).aspx](http://technet.microsoft.com/en-us/library/aa906012(v=sql.80).aspx)

[12] -

<https://www.facebook.com/notes/facebook-engineering/under-the-hood-automated-backups/10151239431923920>

Sitios de alta performance

[13] -

<http://www.channels.com/episodes/show/5265499/Velocity-09-Eric-Schurman-and-Jake-Brutlag-Performance-Related-Changes-and-their-User-Impact->

[14] - <http://stevesouders.com/hpws/rules.php>

[15] -

<https://developer.yahoo.com/blogs/ydn/high-performance-sites-rule-13-configure-etag-7211.html>

[16] - [https://www.facebook.com/note.php?note\\_id=307069903919](https://www.facebook.com/note.php?note_id=307069903919)

Big Data

- [17] - <http://blog-bhaskaruni.blogspot.com/2012/12/facebook-architecture.html>
- [18] - <http://www.fing.edu.uy/inco/cursos/hpc/material/clases/BigData-ML.pdf>
- [19] - <http://hadoop.apache.org>
- [20] - [http://en.wikipedia.org/wiki/Apache\\_Hive](http://en.wikipedia.org/wiki/Apache_Hive)
- [21] - <http://www.happyminds.es/apache-hadoop-introduccion-a-hdfs>
- [22] - <http://hive.apache.org/>
- [23] -  
<http://gigaom.com/2013/11/06/facebook-open-sources-its-sql-on-hadoop-engine-and-the-web-rejoices/>
- [24] - <http://i.stack.imgur.com>
- [25] - [http://www.bidoop.es/casos\\_estudio](http://www.bidoop.es/casos_estudio)
- [26] -  
<http://www.franciscojavierpulido.com/2013/07/bigdata-hadoop-i-definiciones-basicas-y.html>