Integración de búsquedas de texto completo en Bases de Datos NoSQL

Integration of full-text searches NoSQLDatabases

Marvin Ramírez Valenzo[*]
René E. Cuevas Valencia[**]
José Mario Martínez Castro[***]

Fecha de recepción: 4 de marzo del 2011
Fecha de aceptación: 16 de junio del 2011

Resumen

El desarrollo y crecimiento masivo de las redes de computadoras y medios de almacenamiento, a lo largo de los últimos años, ha motivado la aparición de un creciente interés por los sistemas de clasificación automática de documentos. Las bases de datos tradicionales tienen limitaciones a la hora de indexar y realizar búsquedas en tiempo real sobre grandes volúmenes de datos. De hecho, de un poco a esta parte se ha creado una tendencia en lo que es la búsqueda en tiempo real a migrar de bases de datos relacionales a sistemas de índices invertidos. Por tal motivo en este trabajo se pretende usar una arquitectura de búsqueda que se basa en un índice invertido altamente eficiente en una base de datos NoSQL.

Palabras clave: NoSQL, bases de datos, MongoDB, Full Text Search, Lucene, Solr, búsqueda de índice invertido.

Key words: NoSQL, databases, MongoDB, Full Text Search, Lucene, Solr,Bu inverted index search

[*] Universidad Autónoma de Guerrero, Unidad Académica de Ingeniería, Chilpancingo Guerrero; México. Correo electrónico: valenzo@gmail.com

[**] Universidad Autónoma de Guerrero, Unidad Académica de Ingeniería, Chilpancingo Guerrero; México. Correo electrónico: reneecuevas@hotmail.com

[***] Universidad Autónoma de Guerrero, Unidad Académica de Ingeniería, Chilpancingo Guerrero; México. Correo electrónico: jmmtzc@hotmail.com

Introducción

A la llegada del software como servicio, los servicios en la nube de éxito con millones de usuarios, llegaron los problemas de alta escalabilidad a extender el margen de operaciones sin perder calidad. Si bien los modelos de bases de datos relacionales se pueden adaptar para hacerlos escalar incluso en los entornos más difíciles, a menudo, se hacen cada vez menos intuitivos a medida que aumenta la complejidad. Triples y cuádruples JOIN en consultas SQL, a veces, poco eficientes, y sistemas de almacenamiento de resultados en cachés para acelerar la resolución de las peticiones y evitar ejecutar cada vez estas pesadas operaciones, son el pan de cada día en muchos de estos proyectos de software.
Los sistemas NoSQL intentan atacar este problema proponiendo una estructura de almacenamiento más versátil, aunque sea a costa de perder ciertas funcionalidades como las transacciones que engloban operaciones en más de una colección de datos, o la incapacidad de ejecutar el producto cartesiano de dos tablas (también llamadoJOIN).

Las redes sociales le han dado a pensar a los desarrolladores que un sistema gestor de base de datos relacional (SGBDR) no es la infraestructura óptima para un sitio web con decenas de millones de usuarios registrados. Además, los servicios más importantes han liberado sus sistemas de almacenamiento no relacional.

Bases de datos NOSQL

Definición

Las bases de datos NoSQL son sistemas de almacenamiento de información que no cumplen con el esquema entidad-relación. Mientras que las tradicionales bases de datos relacionales basan su funcionamiento en tablas, joins y transacciones ACID. Las bases de datos NoSQL no imponen una estructura de datos en forma de tablas y relaciones entre ellas en ese sentido son más flexibles, ya que suelen permitir almacenar información en otros formatos como clave-valor similar a tablas Hash, Mapeo de Columnas, Documentos o Grafos.

Historia

Originalmente, el término NoSQL, que fue acuñado en 1998, se refería a una base de datos relacional de código abierto que no usaba un lenguaje de consultas SQL (Structured Query Language).

Hasta el 2009, estas cinco letras cayeron en el olvido, pero fue Johan Oskarsson, entonces empleado de Last.fm, quien organizó un evento para tratar las bases de datos distribuidas de código abierto no relacionales, llamándolas "NOSQL", Not-Only SQL.

Características

Las características comunes entre todas las implementaciones de bases de datos distribuidas no relacionales, propietarias o no, suelen ser las siguientes:

- Consistencia eventual: no se implementan mecanismos rígidos de consistencia como los presentes en las bases de datos relacionales, en las cuales la confirmación de un cambio implica una comunicación de este a todos los nodos que lo repliquen. Esta flexibilidad hace que la consistencia se dé, eventualmente, cuando no se hayan modificado los datos durante un periodo. Esto también se conoce como BASE (Basically Available Soft-state Eventual Consistency, o coherencia eventual flexible básicamente disponible), en contraposición a ACID, su analogía en las bases de datos relacionales.

- Estructura distribuida: generalmente, se distribuyen los datos mediante mecanismos de tablas de hash distribuidas (DHT), ya que realmente se trata, según las distintas implementaciones, de redes p2p.
- Escalabilidad horizontal: la implementación típica se realiza en muchos nodos de capacidad de procesado limitado, en vez de utilizar grandes Mainframes.
- Tolerancia a fallos y redundancia (Wikipedia, 2011).

Aplicaciones

Bases de datos documentales:

- CouchDB, de Apache.
- MongoDB, de 10gen MongoDB.
- RavenDB, de Hibernating Rhinos.
- BaseX.
- eXist.
- SimpleDB.
- IBM Lotus Domino.
- Terrastore.

Bases de datos en grafo:

- Neo4j.
- DEX.
- AllegroGraph.
- OrientDB.
- InfiniteGraph.
- Sones GraphDB.
- InfoGrid.
- HyperGraphDB.

Bases de datos clave/valor:

- Cassandra, de Apache The Apache Cassandra.
- BigTable, de Google.
- Dynamo, de Amazon.
- Project Voldemort, de LinkedIn.
- Riak.

Bases de datos multivalor:

- OpenQM.
- Extensible storage engine.

Bases de datos orientadas a objetos:

- db4o.
- GemStone S.
- Objectivity/DB.

Bases de datos tabular:

- HBase, de Apache.
- BigTable, de Google.
- Hypertable.
-

Cuando utilizar una base de datos NoSQL

Si pretendemos desarrollar una aplicación que requiera la lectura/escritura de cantidades de datos y pueda dar servicio a millones de usuarios sin perder rendimiento, entonces, debemos plantearnos el uso de una base de datos NoSQL. Las grandes redes sociales como facebook y twitter o el propio Google las utilizan como medio fundamental de almacenamiento de información. Se puede utilizar una base de datos NoSQL para almacenar toda la información de una aplicación, aunque en la mayoría de los casos, se recurre a sistemas mixtos que combinan los clásicos sistemas relacionales –fácilmente manipulables e interrogables con el lenguaje SQL– con soluciones NoSQL para aquellas funcionalidades que requieren millones de consultas en tiempo real.

Búsquedas de texto completo

Definición

En la recuperación de textos, búsqueda de texto completo se refiere a una técnica para

la búsqueda de un equipo almacenados documento o base de datos . En una búsqueda de texto completo, el motor de búsqueda examina todas las palabras en todos los documentos almacenados en su intento, para que coincida con las palabras de búsqueda suministrados por el usuario. Las técnicas de búsqueda se hicieron comunes en línea, así como las bases de datos bibliográficas en la década de los noventa. Muchos sitios web y programas de aplicación –tales como procesadores de texto de software– ofrecen capacidades de búsqueda de texto completo. Algunos motores de búsqueda web, tales como AltaVista emplean técnicas de búsqueda de texto completo mientras que el índice que otros solo una parte de las páginas web examinadas por su sistema de indexación.

Índice invertido

Un índice invertido –también conocido como archivo de publicaciones o archivo invertido– es una estructura de datos del índice de almacenamiento de una asignación de contenido, tales como palabras o números, a sus ubicaciones en un archivo de base de datos o en un documento o un conjunto de los documentos. El objetivo de un índice invertido consiste en permitir rápido búsquedas de texto completo, a un costo de procesamiento aumenta cuando un documento se agrega a la base de datos. El archivo invertido puede ser la base de datos del propio archivo, en lugar de su índice. Es la estructura de datos más popular utilizado en la recuperación de documentos de sistemas, utiliza a gran escala, por ejemplo, en los motores de búsqueda. Varios importantes de propósito general mainframe, basados en sistemas de gestión de base de datos han utilizado invertido arquitecturas lista, incluyendo DABAS, datos / base de datos y modelo 204.

Hay dos variantes principales de los índices invertidos: un nivel sin precedentes índice invertido (o archivo de índice invertido o simplemente archivo invertido) contiene una lista de referencias a los documentos de cada palabra. Un nivel del índice invertido palabra (o invertida índice o lista invertida) contiene además las posiciones de cada palabra dentro de un documento. Esta última forma ofrece una mayor funcionalidad, por ejemplo, búsquedas de frase, pero necesita más tiempo y el espacio que se creará.

Index

Cuando se trata de un pequeño número de documentos es posible que el texto de búsqueda de lleno a escanear directamente el contenido de los documentos con cada consulta, una estrategia llamada serie de escaneo.

Sin embargo, cuando el número de documentos para buscar es potencialmente de gran tamaño o la cantidad de consultas de búsqueda por realizar es considerable, el problema de búsqueda de texto completo a menudo se divide en dos tareas: la indexación y la búsqueda. La etapa de indexación analizará el texto de todos los documentos y creará una lista de los términos de búsqueda, a menudo llamado un índice, pero más correctamente llamado una concordancia. En la etapa de búsqueda, al realizar una consulta específica, solo el índice se hace referencia en lugar del texto de los documentos originales.

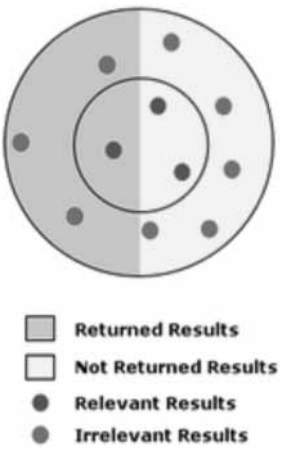
El paso a paso se hace a un registro en el índice para cada término o palabra que se encuentran en un documento y, posiblemente, su posición relativa dentro del documento. Por lo general, en el indizador se ignoran las palabras vacías, como “, la”, que son muy comunes y tiene muy poco sentido ni son útiles para la búsqueda. Algunos también emplean indizadores específico del lenguaje y se derivan de las palabras que son indexa-

das, por ejemplom alguna de las palabras "unidades", "llevó" o "impulsadas" se grabará en el índice bajo un concepto sola palabra "unidad".

La precisión vs recuperar compensación

Este diagrama representa una baja precisión, la búsqueda de bajos recordar cómo se describe en el texto.

Figura 1.



El diagrama representa una baja precisión, y recordatorio de la búsqueda bajo. En el diagrama de la verde y puntos rojos representan la población total de posibles resultados de una búsqueda determinada. Los puntos rojos representan los resultados irrelevantes y los puntos verdes representan los resultados relevantes. La relevancia es indicada por la proximidad de los resultados de búsqueda para el centro del círculo interior. De todos los resultados posibles, se muestran los que fueron devueltos en realidad por la búsqueda y se muestran en una luz azul de fondo. El único ejemplo de un resultado relevante de los tres posibles resultados fue devuelto, por lo que la retirada es una proporción muy baja de 1 / 3 ó 33%. La precisión para el ejemplo es un muy bajo 4,1 ó 25%, ya que solo uno de los cuatro resultados devueltos era pertinente.

Debido a las ambigüedades del lenguaje natural, sistemas de búsqueda de texto completo suele incluir opciones como las palabras vacías para incrementar la precisión y derivados para aumentar memoria. Vocabulario controlado de búsqueda, ayuda a aliviar los problemas de precisión baja por etiquetado documentos, de manera que las ambigüedades se eliminan. El equilibrio entre precisión y exhaustividad es simple: un aumento en la precisión puede disminuir recordar general, mientras que un aumento en el recuerdo disminuye la precisión (Wikipedia, 2011).

Tipos de búsqueda

Restringido campo de búsqueda. Algunos motores de búsqueda les permiten a los usuarios para limitar las búsquedas de texto libre a un determinado campo dentro de un almacenado registro de datos,como "Título" o "Autor".

Consultas booleanas. Proveedores que utilizan booleanos operadores (por ejemplo, "enciclopedia" Y "en línea" NO "Encarta") puede aumentar drásticamente la precisión de una búsqueda de texto libre. El operador dice, en efecto, "no recuperar cualquier documento a menos que contenga estos dos términos". El operador no dice, en efecto, "no recuperar cualquier documento que contiene esta palabra". Si la lista de recuperación recupera muy pocos documentos, el operador OR se puede utilizar para aumentar recordar, considerar, por ejemplo, "enciclopedia" Y "en línea" o "Internet" NO "Encarta". Esta búsqueda recuperará los documentos sobre enciclopedias en línea que utilizan el término "Internet" en lugar de "en línea". Este aumento en la precisión es muy contraproducente, ya que por lo general viene con una dramática pérdida de memoria.

Búsqueda de frase. Una frase de búsqueda incluye solo los documentos que contienen la frase especificada, como "Wikipedia, la enciclopedia libre".

Concepto de búsqueda. Una búsqueda que se basa en la palabra de varios conceptos, por ejemplo, término compuesto de procesamiento . Este tipo de búsqueda se está haciendo popular en el e-Discovery soluciones de muchos.

Búsqueda de concordancia. Una búsqueda de concordancia produce una lista alfabética de todas las palabras principales que se producen en un texto con su contexto inmediato.
Búsqueda de proximidad. Una frase de búsqueda incluye solo los documentos que contienen dos o más palabras separadas por un número determinado de palabras.

Expresión regular. Emplea una potente consulta compleja sintaxis que se puede utilizar para especificar las condiciones de recuperación de manera precisa.

Búsqueda aproximada. Buscará documentos que coinciden con los términos definidos y algunas variaciones en torno a ellos –mediante, por ejemplo, modificar la distancia al umbral de la variación múltiple-
Búsqueda con comodines. Una búsqueda que sustituye a uno o varios caracteres en una consulta de búsqueda para un carácter comodín, como un asterisco . Por ejemplo, utilizar el asterisco en una consulta de búsqueda "s * n" encontrará "pecado", "hijo", "sol", etc. en un texto.

Aplicaciones de busqueda

- Ht-/ / Dig.
- Hyper estraier.
- Lemur / indri.
- Lucene.
- mnoGoSearch.
- Esfinge.
- Swish-e.
- Xapian.
- KinoSearch.

Integración de busquedas de texto completo con NOSQL

Se seleccionó MongoDB porque tiene la combinación de las mejores características de las bases de datos JSON, almacenamiento clave/valor y RDBMSes. Mongo es una base de datos de alto rendimiento, de código abierto y de esquema-libre orientada a documentos (schema-free document-oriented). MongoDB está escrito en C++ y ofrece las siguientes características:

- **Almacenamiento orientado a documento** (la simplicidad y el poder datos de esquemas de tipo JSON).
- **Consultas dinámicas**
- **Soporte completo de índices,** incluido el interior de inner objects y cadenas embebidas.
- **Consulta de perfiles** (Query profiling).
- **Replicación y soporte de fail-over**
- **Almacenamiento eficaz de datos binarios** incluidos objetos grandes (por ejemplo, videos).
- **Auto-sharding** de escalabilidad cloud-level (en la actualidad, en etapa alfa).
- **MapReduce** para agregación compleja Soporte Comercial Disponible

El objetivo fundamental de MongoDB es el de cerrar la brecha existente entre los valores almacenados de clave/valor (key/value) (que son rápidos y altamente escalables) y los sistemas tradicionales RDBMS (que son potentes en funcionalidad).
Se seleccionó Lucene como la herramienta de búsqueda de texto completo, ya que tiene

versiones para otros lenguajes incluyendo-Delphi, Perl, C#, C++, Python, Ruby y PHP, y es útil para cualquier aplicación que requiera indexado y búsqueda a texto completo. Lucene ha sido ampliamente usado por su utilidad en la implementación de motores de búsquedas. Por ello, a veces se confunde Lucene con un motor de búsquedas con funciones de "crawling" y análisis de documentos en HTML incorporadas.

El centro de la arquitectura lógica de Lucene se encuentra el concepto de Documento (Document) que contiene Campos (Fields) de texto. Esta flexibilidad permite a Lucene ser independiente del formato del fichero. Textos que se encuentran en PDF, páginas HTML, documentos deMicrosoft Word, así como muchos otros pueden ser indexados mientras que se pueda extraer información de ellos.

Lucene va mucho más alla que las búsquedas en bases de datos mediante índices FullIndex, permitiéndonos indexar y realizar búsquedas sobre todo tipo de información que pueda representarse de forma textual. Lucene se compone de dos procesos o fases para conseguir realizar búsquedas efectivas: Indexación y Búsqueda (Chodorow y Dirolf, 2010).

MongoDB

MongoDB es:

- Orientado a documentos.
- Los documentos (objetos) se corresponden directamente con los distintos tipos de datos de los lenguajes de programación.
- Los documentos embebidos y los arrays minimizan la necesidad de usar joins.
- Tipados dinámicamente (*schemaless*) para una evolución sencilla del esquema.
- Ausencia de joins y de transacciones (de varios objetos) para lograr un alto rendimiento y una escalabilidad fácil.
- Alto rendimiento.
- Sin joins ni transacciones las lecturas y las escrituras son más rápidas.
- Índices en documentos embebidos y arrays.
- Escrituras asíncronas (opcionales).
- Alta disponibilidad.
- Servidores replicados con mecanismos automáticos de sustitución de un servidor maestro en caso de fallo.
- Fácil escalabilidad.
- Lecturas consistentes eventualmente y distribuidas sobre los servidores replicados.
- *Sharding* automátizado (auto-partición de los datos a lo largo de varios servidores).
- Las lecturas y las escrituras se distribuyen sobre los distintos *shards*.
-

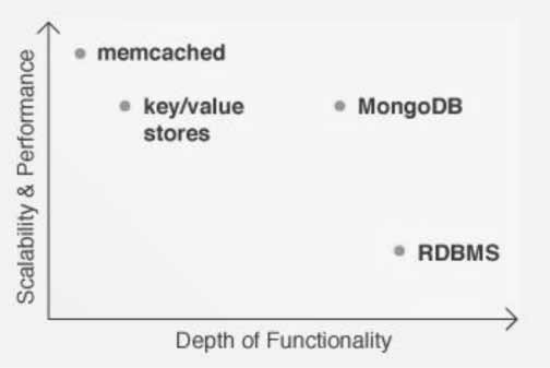
La ausencia de joins y de transacciones hace que las consultas distribuidas sean más fáciles y sencillas. Tiene un lenguaje de consulta muy completo y rico.

Filosofia de MongoDB

Las bases de datos se están especializando: la aproximación "uno es adecuado para todo" ya no se aplica. Por las semánticas de reducción transaccional que la base de datos ofrece, uno puede aún solucionar un interesante conjunto de problemas en los cuales el rendimiento sea muy importante, y la escalabilidad horizatal sea más fácil.

El modelo de datos orientado a documento (JSON) es fácil de codificar, fácil de manejar (sin esquema), y ofrece un excelente rendimiento agrupando internamente los datos relevantes.

Figura 2



Fuente: http://en.wikipedia.org/wiki/Document-oriented_database

Una aproximación no-relacional es el mejor camino para las soluciones de bases de datos con escalabilidad horizontal en muchas máquinas. Mientras hay una oportunidad para relajar ciertas capacidades para el mejor rendimiento, también hay una necesidad para una funcionalidad profunda que ofrezca almacenamientos puros clave/valor. La tecnología de base de datos se debería ejecutar en cualquier parte, estando disponibles para su ejecución tanto en tus propios servidores como en máquinas virtuales. así como también como un servicio en nube de paga por lo que uses (mongoDB, 2011).

Caso de uso

Mejor adaptado. Almacenamiento de datos operacionales de un sitio web. MongoDB es muy bueno para inserciones, actualizaciones y consultas en tiempo real. Se provee escalabilidad y replicación, para las cuales son necesarias funciones para almacenar datos en tiempo real para sitios web grandes. Algunos casos específicos: gestión de contenidos almacenamiento de comentarios, gestión, votos contadores de página en tiempo real registro de usuario, perfil, datos de sesión.

Cacheo. Con su potencial para alto rendimiento, MongoDB trabaja bien como capa de chacheo en una infraestructura de información. El respaldo de caché persistente de MongoDB asegura que en un sistema rearranque la capa de datos hacia abajo sin estar abrumado con la actividad de crecimiento de la caché.
Alto volumen, bajo valor de datos. Los problemas en los cuales un DBMS tradicional podría ser costoso para los datos en cuestión. En muchos casos, los desarrolladores deberían escribir tradicionalmente código personalizado para un sistema de ficheros en lugar de usar ficheros planos u otras metodologías.
Los problemas que requieran alta escalabilidad. Mongo está bien adaptado para problemas en los cuale la base de datos debe comprimir decenas o cientos de servidores. La integración del motor Map/Reduce está planificado en la ruta de Mongo.

Almacenamiento de objetos de programa y datos JSON (y equivalentes). El formato BSON de Mongo hace muy sencillo almacenar y recuperar datos en un formato de estilo documento / "sin esquema". Añadir nuevas propiedades a los objetos existentes es fácil y no requiere operaciones de bloqueo del estilo de "ALTER TABLE".

Menos adaptado. Sistemas altamente transaccionales, tales como sistemas de banca y contabilidad. MongoDB, como la mayor parte de soluciones NOSQL, ofrece transaccionalidad ligera: atomicidad únicamente sobre documentos sencillos. Las aplicaciones con transacciones altamente complejas son mejor resueltas por un RDBMS tradicional.

Business Intelligence tradicional. Los Data Warehouses están mejor adaptados para problemas específicos de bases de datos BI. Sin embargo, MongoDB puede trabajar bien para algunos problemas de reporting y analíticas, en los cuales los datos están previa-

mente destilados o agregados en tiempo de ejecución. Por último, están los problemas que requieran SQL.

Lucene

Introducción

Lucene es una novedosa herramienta que permite tanto la indexación como la búsqueda de documentos. Fue creada bajo una metodología orientada a objetos e implementada completamente en Java, no se trata de una aplicación que pueda ser descargada, instalada y ejecutada, sino de una API flexible, muy potente y realmente fácil de utilizar, mediante la cual se pueden añadir, con pocos esfuerzos de programación, capacidades de indexación y búsqueda a cualquier sistema que se esté desarrollando. Originalmente, fue escrita por Doug Cutting, en septiembre del 2001, y pasó a formar parte de la familia de código abierto de la fundación Jakarta. Desde entonces, debido a su mayor disponibilidad, ha atraido a un gran número de desarrolladores, incluso, empresas como Hewlett Packard, FedEx, etc.

A parte de Lucene, existen otras herramientas que permiten realizar la indexación y la búsqueda de documentos, pero dichas herramientas han sido optimizadas para usos concretos, lo cual implica que intentar adaptar dichas herramientas a un proyecto específico sea una tarea realmente difícil. La idea que engloba Lucene es completamente diferente, ya que su principal ventaja es su flexibilidad, que permite su utilización en cualquier sistema que lleve a cabo procesos de indexación.

Características

A continuación, se detallan algunas características que hacen de Lucene una herramienta flexible y adaptable de indexación incremental vs indexación por lotes. El término de indexación por lotes se utiliza para referirse a aquellos procesos de indexación, en los cuales, una vez que ha sido creado el índice para un conjunto de documentos, el hecho de intentar añadir algunos documentos nuevos es una tarea difícil por lo que se opta por reindexar todos los documentos de nuevo. Sin embargo, en la indexación incremental se pueden añadir documentos a un índice ya creado con anterioridad de forma fácil. Lucene soporta ambos tipos de indexación.

Origen de datos. Muchas herramientas de indexación solo permiten indexar ficheros o páginas web, lo que supone un serio inconveniente cuando se tiene que indexar contenido almacenado en una base de datos.

Contenido etiquetado. Algunas herramientas, tratan los documentos como simples flujos de palabras. Pero otras, como Lucene, permiten dividir el contenido de los documentos en campos y así poder realizar consultas con un mayor contenido semántico. Esto es, se pueden buscar términos en los distintos campos del documento concediéndole más importancia según el campo en el que aparezca. Por ejemplo, si se dividen los documentos en dos campos, título y contenido, se puede conceder mayor importancia a aquellos documentos que contengan los términos de la búsqueda en el campo título.

Técnica de indexación. Existen palabras como a, unos, el, la, etc., que añaden poco significado al índice, son palabras poco representativas del documento. Al eliminar estas palabras del índice, se reduce considerablemente el tamaño de este, así como el tiempo de indexación. Estas palabras están contenidas en lo que se denomina lista de parada, que es la técnica de indexación contemplada por Lucene.

Concurrencia. Lucene gestiona que varios usuarios puedan buscar en el índice de forma simultánea, así como que un usuario mo-

difique el índice al mismo tiempo que otro lo consulta.

Elección del idioma. Tal y como ya se indicó con anterioridad Lucene trabaja con listas de parada, las cuales son proporcionadas por el desarrollador que está utilizando Lucene, esto permite escoger el idioma que se va a utilizar.

Indexación. El proceso de indexación consiste en analizar y extraer de entre todo lo disponible, lo verdaderamente relevante. Posteriormente, con esa información se crea el índice a partir del cual se realizarán las búsquedas. El índice es una estructura de datos que permite acceso rápido a la información, algo similar semánticamente a lo que podría ser el índice de un libro (Solr, 2011).

Soluciones basadas en Lucene

SOLR. Es de código abierto, plataforma de búsqueda a partir del proyecto Apache Lucene. Sus características principales incluyen la búsqueda de gran alcance de texto, resaltado de búsqueda de facetas, la agrupación dinámica, la integración de bases de datos, documento enriquecido (por ejemplo, Word, PDF), manipulación y búsqueda geoespacial. SOLR es altamente escalable, proporcionando búsqueda distribuido y replicación de índices, así como las competencias de las funciones de búsqueda y navegación de muchos de los sitios más grandes del mundo Internet.

SOLR está escrito en Java y se ejecuta como una búsqueda de texto completo del servidor independiente dentro de un contenedor de servlets como Tomcat. SOLR utiliza el Java Lucene, la colección de la búsqueda en su base para la indización de texto completo y de búsqueda, y tiene como HTTP REST / XML y JSON APIs que hacen que sea fácil de usar desde prácticamente cualquier lenguaje de programación. Los externos de configuración SOLR poderoso le permite adaptarse a casi cualquier tipo de aplicación sin codificación Java, y tiene una amplia arquitectura de plugin cuando más avanzadas de personalización es necesario.

Elastic Search. Elastic Search es un motor de búsqueda en tiempo real y distribuido que se diferencia de los demás por hacer pública la gestión y consulta de estos índices mediante un API REST, lo que hace realmente sencillo comenzar a trabajar con el motor sin siquiera tener conocimientos de ningún lenguaje de programación.

Además del API REST, existen API en Java y Groovy. Soporta Sharding y Multi-Tenancy –un mismo índice dividido en partes automáticamente vs. generación de divisiones lógicas sobre un mismo índice–. Está liberado bajo licencia Apache, y su simplicidad realmente impresiona. El creador es Shay Banon, exdirector de tecnología de GigaSpaces y fundador del proyecto Compass. En su blog Shay Banon describe a Elastic Search como todo lo que él quiso que fuera Compass 3.0 y nunca pudo ser.

Katta.

Katta es una librería, no demasiado conocida, basada en Hadoop y Lucene. Básicamente realiza sharding sobre los índices de Lucene sirviendo de este modo un único índice repartido entre múltiples servidores. Tiene licencia Apache y ofrece acceso en tiempo real a los índices además de encargarse automáticamente del sharding y la tolerancia a fallos. Apartándonos ya de los temas técnicos, parece que la mayor actividad en las listas de correo se registró en 2009 decayendo un poco en los últimos años. Seguramente el haber ganado en estabilidad y el hecho

de que no sea una librería no demasiado conocida haya contribuido a esto. Podéis ver más sobre Katta, aquí..

Zoie.

Zoie es el sistema de indexado y búsqueda en tiempo real de Linkedin. Linkedin lo donó como proyecto Open Source en 2008 y su sitio web sigue utilizando, gestionando millones de búsquedas diariamente. Se trata de una modificación de Lucene adaptándola a los requisitos de Linkedin sin incluir funcionalidades como sharding, tolerancia a fallos, etc. que sería necesario añadir.

Creación de la aplicación

Se desarrolló una aplicación con SOLR y MongoDB porque son rápidos y fáciles de trabajar. Se ha creo una función para la conexión entre MongoDB y Solr, con un insert () que toman un dataRecord (leer el archivo) como un argumento.

```
public function insert(DataRecord $record) {
$this->collection->update(array('id' => $re-
cord->id() ),
array('$set' => array(
'list.' . $record->year() => $record-
>getDetails()->toArray())),
array('upsert' => true));
}
```

}Se trata de introducir un documento de la colección si no está allí. Cuando se sí está, se agrega un elemento a (la lista 'elemento anidado) »con el valor $record->year() como un elemento clave. El valor será el valor de $record->getDetails() . El toArray () llamada está allí porque el conductor mongo espera matrices para almacenar. Si la clave existe, solo se actualizará con los datos de los detalles del objeto. Una revisión del documento en SOLR, un SolrInputDocument como argumento. El punto fino es que al indexar lo que tienes que leer es el objeto de completar la base de datos, a fin de obtener todos los datos. El dataRecord que fue leído desde el archivo y se almacena con Upsert solo puede haber sido parte de la imagen. Fue una consecuencia de la estructuración de los datos como una colección de objetos persona en MongoDB, en lugar de una larga lista de registros en la versión anterior. Esta mapas mejor el dominio.

```
public function index(SolrInputDocument
$document) {
$response = $this->solr-
>addDocument($document);
 if($this->pendingDocuments++ == $this-
>commitInterval) {
$this->commit();
$this->pendingDocuments = 0;
}
return $response->getResponse();
 }
```

Pequeñas pruebas indican cuatro minutos para la indexación de documentos con 6000 registros y 17 segundos con un compromiso cada documento de 2000 –y al final, por supuesto–. El código anterior se compromete cada $ commitInterval (10.000 por defecto) para acelerar las cosas un poco. Tenga en cuenta también que el commit () y optimizar () llama a SOLR y puede agotar el tiempo, ya que puede tomar un tiempo para finalizar. SOLR no es el tiempo de espera, sino el servidor de aplicaciones Java que se está ejecutando. Cuando esto sucede se produce una excepción en el controlador de php que tiene que ser capturados (Lucene, 2011).

Resultados

La hora de la inserción de los lotes de actualización:

- Intervalo de indexación para Solr: 10.000.
- Sctualización de registro de SOLR desactivado (por defecto es muy detallado).
- nssize = 1024 para MongoDB.

Uso de espacio

No hay preasignación, se hizo por MongoDB por lo que creó los archivos de datos según sea necesario. Esto significa que el último fue creado en 2GB y muy bien puede ser casi vacío. Mongo crea archivos de una manera doble, pasando de 64 MB a 2G como esta: 64, 128, 256, 512, 1GB, 2GB.

Tabla 1

Sistema de	Índice	Datos
Mongo + SOLR inicial de importación	744 MB	1294 MB (3 GB de archivos de datos)
Mongo + SOLR inicial + un conjunto de datos adicionales.	1538 MB	4348 MB

Uso de la CPU

MongoDB y Solr ocupa 35-40% de la CPU y alrededor de 10 MB de RAM. MongoDB está utilizando aproximadamente el 10% de la CPU (con 1,1 GB de RAM) y Solr (Tomcat, que lo es) es el gasto del 30% y en promedio 300 MB de RAM.

Conclusiones

Cuando utilizar una base de datos NoSQL

(ABRE CITA) Si pretendemos desarrollar una aplicación que requiera la lectura/escritura de cantidades de datos y pueda dar servicio a millones de usuarios sin perder rendimiento, entonces debemos plantearnos el uso de una base de datos NoSQL. Las grandes redes sociales como facebook y twitter o el propio Google las utilizan como medio fundamental de almacenamiento de información. Se puede utilizar una base de datos NoSQL para almacenar toda la información de una aplicación, aunque en la mayoría de los casos se recurre a sistemas mixtos que combinan los clásicos sistemas relacionales (fácilmente manipulables e interrogables con el lenguajeSQL) con soluciones NoSQL para aquellas funcionalidades que requieren millones de consultas en tiempo real. (CIERRA CITA)

Búsqueda de información

Todos los productos NoSQL proporcionan, de una forma u otra, la capacidad de datos de consulta. Algunos utilizan SQL, como consultas, otros utilizan mapas. Sin embargo, en un momento dado, independientemente de dónde o cómo se almacenan los datos, se quiere añadir la búsqueda a esta. Las bases de datos NoSQL nos dan la solución a los problemas de almacenamiento de datos y escalibilidad , pero es importante poder busca en tiempo real el mundo de infomaciòn que se pueden guardar en la bases de datos NoSQL.

La solución que proponemos para los requisitos de escala de nuestro almacenamiento de datos, es añadir un sistema de búsqueda invertida. Y la solución de las necesidades de búsqueda a escala. El escenario ideal es que nuestra solución de búsqueda nos permita buscar en todos nuestros datos.

En una indexación de procedimiento almacenado ejecuta dentro de nuestro grupo NoSQL y cada vez que algo cambia, el mis-

mo cambio se aplica al índice de busqueda, la conversión de cualquier NOSQL con formato JSON , de almacenamiento basados en la columna, o incluso de objetos de base es solo una cuestión de convertirlos en una indexables JSON porque las búsquedas de datos es flexible la conversión es extremadamente simple.

Al hacer la integración de busqueda invertida con NoSQL, realmente podemos obtener un alto rendimiento, baja latencia, la red de datos escalables con la opción de realizar búsquedas de texto completo en todas las parte de los datos almacenados (Chodorow, 2010).

Referencias

Chodorow, K. y Dirolf, M.(2010), *MongoDB: The Definitive Guide. O'Reilly Media,* O'Reilly & Associates, Inc.

Chodorow, K. (2010), Escaling MongoDB. O'Reilly Media. O'Reilly & Associates, Inc

Lucene. (2011), Recuperado de: http://lucene.apache.org/

MongoDB (2011). http://www.mongodb.org/

SOLR (2011). Recuperado de: http://lucene.apache.org/solr/

Wikipedia (2011). Recuperado de: http://es.wikipedia.org/