

INFORME DEL PROYECTO E INVESTIGACION DE TECNOLOGIAS DISRUPTIVAS

Participantes

ING. ESP. JOSE JAVIER VAQUIRO LADINO
CC. No. 14885455

ING. JOSÉ HERNANDO BENAVIDES
CC. No. 12.996.632

Tutor

Ing. Roberto Mauricio Cárdenas

Universidad Nacional Abierta y a Distancia - UNAD

Tendencias Disruptivas en TI

Maestría en Gestión de TI

2016

Contenido

1.	NOMBRE DEL PROYECTO.....	3
2.	PLANTEAMIENTO DEL PROBLEMA	3
3.	OBJETIVOS	3
3.1.	GENERALES.	3
4.	INTRODUCCIÓN.	4
5.	VIDEO INFORME DEL PROYECTO	5
6.	MOTIVADORES DE NEGOCIO Y PROBLEMA.....	5
6.1.	Problema.	5
6.2.	Motivadores del Negocio.....	5
7.	DISEÑO DE LA SOLUCION INTEGRADORA	6
8.	DISRUPTIVAS Y CAMPOS TECNOLÓGICOS.....	6
8.1.	Internet de Las Cosas (IoT): Rasberry Pi.....	6
8.2.	Procesamiento de Datos en Batch : Spark, Hadoop.....	7
8.3.	Buscadores Empresariales: Solr, ElasticSearch, Lucene.....	9
8.4.	Procesamiento de Datos en Streaming: Storm, Spark Streaming	10
8.5.	Análisis de datos con Grafos: GraphX Spark, Giraph.....	15
8.6.	Almacenamiento en pares clave – valor: Redis, Riak, Memcached	15
8.7.	Almacenamiento en columnas – Bigtable, Cassandra, HBase, Dynamo	18
8.8.	Almacenamiento en motores documentales – MongoDB, CouchDB	19
8.9.	Almacenamiento en grafos – Neo4j, Titan, OrientDB	22
8.10.	Computación distribuida con contenedores – Docker, Mesos	23
8.11.	Programación Reactiva: Scala(Akka), Java Rx.....	27
8.12.	Arquitectura Web de Micro servicios: NodeJs, Dropwizard, Spring Boot,etc.	27
8.13.	HTML5: Componentes Web, Frameworks, Web Sockets.	29
9.	CONCLUSIONES	30
10.	REFERENCIAS BIBLIOGRÁFICAS.....	32

1. NOMBRE DEL PROYECTO

EDUKAR

DESARROLLO E IMPLEMENTACIÓN DE UNA PLATAFORMA VIRTUAL PARA LA GESTIÓN Y ADMINISTRACIÓN DE PROCESOS EDUCATIVOS, DIRIGIDO A TODA LA COMUNIDAD ACADÉMICA DE LA INSTITUCION EDUCATIVA “EL SOCORRO” UBICADA EN SAN JUAN DE PASTO-NARIÑO.

2. PLANTEAMIENTO DEL PROBLEMA

En la actualidad en la institución educativa municipal "El socorro", ubicada en la ciudad de San Juan de Pasto, Nariño, existe una situación limitante con la gestión y manejo de los procesos académicos, dirigidos a toda la comunidad educativa de dicha Institución.

3. OBJETIVOS

3.1. GENERALES.

- ▶ Solucionar una necesidad Académica con los procesos de Enseñanza-Aprendizaje en la mayor parte de instituciones educativas públicas y privadas de la región del suroccidente colombiano.
- ▶ Mejorar los procesos académicos y administrativos en la formación académica para toda la comunidad educativa en los diferentes grados de las instituciones públicas y privadas de la región del suroccidente colombiano.

4. INTRODUCCIÓN.

El presente trabajo tiene como propósito dar a conocer las diferentes herramientas de software que hacen parte de la tecnología disruptiva. Tomando como referencia que la tecnología disruptiva es un proceso por el cual un producto o servicio se abre camino, inicialmente con aplicaciones sencillas con los clientes de baja gama y luego dichas aplicaciones pueden mejorar hasta desplazar competidores establecidos.

En primer lugar, se da a conocer algunos conceptos fundamentales sobre diversos campos tecnológicos de gran Disrupción como Frameworks, computación distribuida, Arquitectura web, gestores de Bases de Datos, entre otros. Luego se expone el uso de algunos campos fundamentados a través de un video y un Blog relacionado.

En segunda instancia, brindamos la información relacionada con nuestro proyecto de grado, como se mencionó anteriormente, este consiste en el **“Desarrollo e Implementación de una Plataforma Virtual para la gestión y administración de Procesos Educativos dirigido a las instituciones públicas de educación básica media”**. Nuestra metodología de trabajo como solución integradora, se fundamentó en un diseño de la Interfaz Gráfica de Usuario (GUI) y la construcción de un Blog, en el cual se registró toda la información correspondiente a nuestro proyecto.

En virtud de lo anterior, cabe justificar que la innovación es fundamental para la evolución y el progreso de la sociedad. Las organizaciones que se oponen al cambio están condenadas a desaparecer ante aquellas que introducen innovaciones en los distintos campos de actividad social. Las organizaciones, especialmente cuanto más complejas son, mantienen fuertes resistencias a cambiar para adaptarse a los nuevos retos y entornos.

5. VIDEO INFORME DEL PROYECTO

Link del Video

https://www.youtube.com/watch?v=XbAoB5S9jUk&feature=em-upload_owner

6. MOTIVADORES DE NEGOCIO Y PROBLEMA

6.1. Problema.

En la actualidad en algunas instituciones educativas públicas y privadas, existe una situación limitante con la gestión y manejo de los procesos académicos dirigidos a toda la comunidad educativa de la región del sur occidente colombiano.

6.2. Motivadores del Negocio.

- ▶ Lograr una efectividad del aprendizaje de los estudiantes en gran parte de la población de la región del suroccidente colombiano.
- ▶ Solucionar una necesidad Académica y Administrativa en la mayor parte de instituciones educativas públicas y privadas de la región del suroccidente colombiano.
- ▶ Usar todas las herramientas tecnológicas ofrecidas por las nuevas tendencias disruptivas al servicio de la formación académica en las diferentes instituciones públicas y privadas de la región del suroccidente colombiano.

- ▶ Mejorar los procesos académicos y administrativos en la formación académica para toda la comunidad educativa en los diferentes grados de las instituciones públicas y privadas de la región del suroccidente colombiano.
- ▶ Brindar a toda la comunidad educativa y administrativa una Interfaz intuitiva y amigable para llevar a cabo el desarrollo de todos los procesos académicos y de Enseñanza-Aprendizaje.
- ▶ Implementar una herramienta Didáctica activa que permita el apoyo a los docentes planear y ejecutar los procesos de Enseñanza-Aprendizaje en todos los grados de las diferentes instituciones educativas la región del suroccidente colombiano.

7. DISEÑO DE LA SOLUCION INTEGRADORA

<http://tdisruptivas.blogspot.com.co/>

8. DISRUPTIVAS Y CAMPOS TECNOLÓGICOS

8.1. Internet de Las Cosas (IoT): Raspberry Pi

La amplia variedad de dispositivos conectados, que se conoce como el “Internet de las Cosas” (IoT), está produciendo una diversidad de nuevos datos provenientes de los sensores que están contenidos en dichos dispositivos. Estos datos prometen nuevos servicios, una eficiencia mejorada y, posiblemente, modelos de negocio más competitivos.

La Raspberry Pi es un equipo pequeño y asequible de una placa que va a utilizar para diseñar y desarrollar dispositivos IO divertidos y prácticos, mientras que el aprendizaje de la programación y el hardware del equipo.

8.2. Procesamiento de Datos en Batch : Spark, Hadoop

Spark.

Spark es una plataforma informática de código abierto desarrollado en Scala, para realizar análisis y procesos avanzados. Desde un principio, Spark fue diseñado para soportar en memoria algoritmos iterativos que se pudiesen desarrollar sin escribir un conjunto de resultados cada vez que se procesaba un dato. Esta habilidad para mantener todo en memoria es una técnica de computación de alto rendimiento aplicado al análisis avanzado, la cual permite que Spark tenga unas velocidades de procesamiento que sean 100 veces más rápidas que las conseguidas utilizando MapReduce.

Spark tiene un framework integrado para implementar análisis avanzados que incluye la librería MLlib, el motor gráfico GraphX, Spark Streaming, y la herramienta de consulta Shark. Esta plataforma asegura a los usuarios la consistencia en los resultados a través de distintos tipos de análisis.

Spark es conocido como un framework liviano en Java. Considerado también como un micro framework inspirado en Sinatra para la creación de **aplicaciones web en Java** de forma rápida y con el mínimo esfuerzo. Spark está escrito en Scala Programming Language y se ejecuta en Java Virtual Machine (JVM), se centra en desarrolladores que deseen escribir sus aplicaciones sólidas en Java, de la forma más simple y directa posible, y sin necesidad de complicados archivos de configuración (XML normalmente) como lo hacen algunos frameworks más completos que existen en la actualidad, como Spring, aunque cada día esta tarea se hace más sencilla. Por otro lado, también es un paradigma totalmente diferente en comparación con el uso (a veces excesivo) de anotaciones, como se puede ver en JAX-RS por ejemplo, para llevar a cabo muchas veces tareas relativamente sencillas.

Características de SPARK

- Soporta más que un mapa y reducir funciones.
- Optimiza gráficos operador arbitrarias.
- La evaluación perezosa de consultas de datos grandes que ayuda con la optimización del flujo de trabajo global de procesamiento de datos.
- Proporciona APIs concisas y consistentes en Scala, Java y Python.
- Ofrece shell interactivo para la Scala y Python. Esto aún no está disponible en Java.

//Un ejemplo de aplicación web con Spark:

```
public class HelloWorld {

    public static void main(String[] args) {

        get(new Route("/hello") {
            @Override
            public Object handle(Request request, Response response) {
                return "Hello World!";
            }
        });
    }
}
```

HADOOP

Hadoop es un sistema de código abierto que se utiliza para **almacenar, procesar y analizar** grandes volúmenes de datos (Big Data). Hadoop se conoce como una gran tecnología de procesamiento de datos ha sido de alrededor de 10 años y ha demostrado ser la solución de elección para el procesamiento de grandes conjuntos de datos.

MapReduce es una gran solución para los cálculos de un solo paso, pero no es muy eficaz para los casos de uso que requieren cálculos de múltiples pasadas y algoritmos. Cada paso en el flujo de trabajo de procesamiento de datos tiene una fase Mapa y uno Reducir fase y tendrás que convertir cualquier caso de uso en modelo MapReduce para aprovechar esta solución.

Los datos de salida de trabajo entre cada paso tienen que ser almacenado en el sistema de archivos distribuido antes de comenzar el siguiente paso. Por lo tanto, este enfoque tiende a ser lento debido a la replicación y de almacenamiento en disco. Además, las soluciones Hadoop típicamente incluyen grupos que son difíciles de configurar y administrar. También se requiere la integración de varias herramientas para diferentes casos de uso de datos grandes (como Mahout de aprendizaje automático y la tormenta para el streaming de procesamiento de datos).

Esta circunstancia comporta que, aquella información que antes las empresas no podían procesar debido a los límites de la tecnología existente o a barreras de tipo económico, que se hacían insalvables en muchos casos; hoy pueda ser almacenada, gestionada y analizada, gracias a Hadoop.

Cualquier organización que utilice Hadoop puede obtener información nueva, al mismo tiempo que descubre y aplica cualquier otro tipo de **análisis a sus datos**, como por ejemplo una regresión lineal sobre millones de registros de su histórico.

Ventajas

- ▶ Aísla a los desarrolladores de todas las dificultades presentes en la programación paralela.
- ▶ Cuenta con un ecosistema que sirve de gran ayuda al usuario, ya que permite distribuir el fichero en nodos, que no son otra cosa que ordenadores con commodity-hardware.
- ▶ Es capaz de ejecutar procesos en paralelo en todo momento.
- ▶ Dispone de módulos de control para la monitorización de los datos.
- ▶ Presenta una opción que permite realizar consultas.
- ▶ También potencia la aparición de distintos add- ons, que facilitan el trabajo, manipulación y seguimiento de toda la información que en él se almacena.

8.3. Buscadores Empresariales: Solr, Elasticsearch, Lucene

Solr: es un motor de búsqueda de [código abierto](#) basado en la [biblioteca Java](#) del proyecto [Lucene](#), con APIs en [XML/HTTP](#) y [JSON](#), resaltado de resultados, [búsqueda por facetas](#), [caché](#), y una interfaz para su administración.

ElasticSearch: es un buscador distribuido en tiempo real flexible y potente y un motor de análisis de código abierto, más popular para entornos empresariales.

Lucene: es una API de código abierto para recuperación de información, originalmente implementada en Java por Doug Cutting. Está apoyado por el Apache Software Foundation y se distribuye bajo la Apache Software License. Lucene tiene versiones para otros lenguajes incluyendo Delphi, Perl, C#, C++, Python, Ruby y PHP.

8.4. Procesamiento de Datos en Streaming: Storm, Spark Streaming

Los datos de Streaming son datos que se generan constantemente a partir de miles de fuentes de datos, que normalmente envían los registros de datos simultáneamente en conjuntos de tamaño pequeño (varios kilobytes). Los datos de Streaming incluyen diversos tipos de datos, como archivos de registros generados por los clientes que utilizan sus aplicaciones móviles o web para: compras electrónicas, información de redes sociales, operaciones bursátiles o servicios geoespaciales, así como telemetría de dispositivos conectados o instrumentación en centros de datos.

Estos datos deben procesarse de forma secuencial y gradual registro por registro o en ventanas de tiempo graduales, y se utilizan para una amplia variedad de tipos de análisis, como correlaciones, agregaciones, filtrado y muestreo. La información derivada del análisis aporta a las empresas visibilidad de numerosos aspectos del negocio y de las actividades de los clientes, como el uso del servicio (para la medición/facturación), la actividad del servidor, los clics en un sitio web y la ubicación geográfica de dispositivos, personas y mercancías, y les permite responder con rapidez ante cualquier situación que surja. Por ejemplo, las empresas pueden supervisar los cambios en la opinión pública de sus marcas y productos al analizar constantemente las transmisiones de los medios sociales.

Ventajas

- ▶ El procesamiento de los datos de Streaming supone una ventaja en la mayoría de las situaciones en las que se generan datos nuevos y dinámicos de forma constante.
- ▶ Es apto para la mayoría de los sectores y casos de uso de Big data.

Ejemplos de Procesamiento de Datos en Streaming.

- ▶ Los sensores de los vehículos de transporte, el equipo industrial y la maquinaria agrícola envían datos a una aplicación de Streaming. La aplicación supervisa el rendimiento, detecta cualquier posible defecto de forma anticipada y envía el pedido de un recambio automáticamente, lo que evita el tiempo de inactividad del equipo.
- ▶ Una institución financiera controla los cambios en la bolsa en tiempo real, procesa el valor en riesgo y modifica las carteras automáticamente en función de los cambios en los precios de las acciones.
- ▶ Un sitio web inmobiliario controla un subconjunto de datos de los dispositivos móviles de los clientes y realiza recomendaciones en tiempo real acerca de los inmuebles que deben visitar en función de su ubicación geográfica.
- ▶ Una compañía de juegos en línea recopila datos de **Streaming** acerca de las interacciones de los jugadores con el juego y los envía a su plataforma de juegos. A continuación, analiza los datos en tiempo real, ofrece incentivos y experiencias dinámicas que involucran a los jugadores.

SPARK STREAMING:

Spark Streaming se puede utilizar para procesar los datos de Streaming en tiempo real. Esto se basa en el estilo por lotes micro de la informática y de procesamiento. Utiliza la DSTREAM que es básicamente una serie de DDR, para procesar los datos en tiempo real. Spark Streaming puede ingerir datos de un amplio de fuentes, incluyendo flujos provenientes de Apache Kafka, Apache Flume, Amazon Kinesis y Twitter, así como de sensores y dispositivos conectados por medio de sockets TCP. También se pueden procesar datos almacenados en sistemas de archivos como HDFS o Amazon S3.

Spark Streaming puede procesar datos utilizando una variedad de algoritmos y funciones tales como map, reduce, join y Windows. Una vez procesados, los datos son enviados a archivos en file systems o para popular dashboards en tiempo real.

*A grandes rasgos, lo que hace **Spark Streaming** es tomar un flujo de datos continuo y convertirlo en un flujo discreto, llamado **DStream**, formado por paquetes de datos.*

Internamente, lo que sucede es que Spark Streaming almacena y procesa estos datos como una secuencia de RDDs (Resilient Distributed Data). Un RDD es una colección de datos particionada (distribuida) e inmutable. Es la unidad de información que el motor de procesamiento de Spark (Spark Core) tradicionalmente consume. Así que cuando usamos Spark Streaming para alimentar un stream a Spark Core, éste último los analiza de forma normal, sin enterarse de que está procesando un flujo de datos, porque el trabajo de crear y coordinar los RDDs lo realiza Spark Streaming.

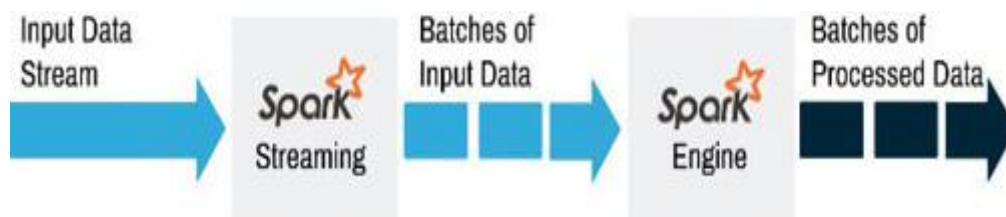


Figura 1. Spark Streaming prepara un flujo para que Spark Core lo pueda consumir.



Figura 2. Spark Streaming genera RDDs por intervalos de tiempo.

STORM.

Apache Storm, es un sistema que sirve para recuperar Streams de datos en tiempo real desde múltiples fuentes de manera distribuida, tolerante a fallos y en alta disponibilidad. Storm está principalmente pensado para trabajar con datos que deben ser analizados en tiempo real, por ejemplo, datos de sensores que se emiten con una alta frecuencia o datos que provengan de las redes sociales donde a veces es importante saber qué se está compartiendo en este momento.

Se compone de dos partes principalmente. La primera es la que se denomina Spout y es la encargada de recoger el flujo de datos de entrada. La segunda se denomina Bolt y es la encargada del procesado o transformación de los datos.

Componentes de Storm.

Un clúster de Storm es muy parecido a un clúster Hadoop. El equivalente al Job MapReduce en Hadoop sería el concepto de topología en Storm. La principal diferencia es que mientras un Job MapReduce termina cuando finaliza la tarea, una topología se queda esperando datos de entrada eternamente, mientras no mates el proceso claro está.

La arquitectura de Storm, se divide en los siguientes componentes:

- ▶ El **master node**; ejecuta el demonio llamado Nimbus responsable de distribuir el código a través del clúster (similar al JobTracker de Hadoop). Realiza también la asignación y monitorización de las tareas en las distintas máquinas del clúster.
- ▶ Los **worker nodes**; ejecutan el demonio Supervisor encargado de recoger y procesar los trabajos asignados en la máquina donde corre. Estos nodos ejecutan una porción de la topología para que así se puedan distribuir los trabajos a lo largo del clúster. Si fallara un worker node el demonio Nimbus se daría cuenta y redirigiría el trabajo a otro worker node.
- ▶ **Zookeeper**; aunque no es un componente como tal de Storm, sí que es necesario montar un Apache Zookeeper ya que será el encargado de la coordinación entre el Nimbus y los Supervisors. También es el encargado de mantener el estado ya que el Nimbus y los Supervisors son stateless.

Spout.

El componente Spout de Storm es el encargado de la ingesta de los datos en el sistema, por ejemplo, si tenemos que leer un fichero de texto y contar las palabras, el componente que recibiría los Streams del fichero sería el Spout. Otro típico ejemplo sería Twitter. Si queremos recoger determinados tweets para posteriormente procesarlos o realizar algún tipo de analítica sobre ellos, el encargado de conectar con el API de Twitter y recoger los datos sería el Spout.

Bolt.

El Bolt es encargado de consumir las tuplas que emite el Spout, las procesa en función de lo que dicte el algoritmo que programamos sobre los streams de entrada y puede emitirlos a otro Bolt. Es recomendable que cada Bolt realice una única tarea. Si necesitamos realizar varios cálculos o transformaciones sobre los datos que le llegan al Bolt, lo mejor es que se dividan en distintos Bolt para mejorar la eficiencia y la escalabilidad. Tanto el Spout como el Bolt emiten tuplas que serán enviadas a los Spout que estén suscritos a ese determinado stream configurado en la topología. Por ejemplo, si queremos contar las veces que aparece cada palabra en un texto, podemos hacer un Bolt que se encargue de contar las que empiezan por vocal y otro para las que empiezan por consonante. El Spout sería el encargado de redirigir a uno u otro Bolt

Topología.

Una topología en Storm es similar a un grafo. Cada nodo se encarga de procesar una determinada información y le pasa el testigo al siguiente nodo. Esto se configura previamente en la topología. La topología se compone de Spots y Bolts. El modo clúster de Storm como su propio nombre indica, ejecuta la topología en clúster, es decir distribuye y ejecuta nuestro código en las distintas máquinas. Es el considerado modo producción.

Modos de funcionamiento.

Storm puede funcionar en dos modos: local y clúster. El modo local es muy útil para probar el código desarrollado en la topología de Storm ya que corre en una única JVM por lo que podemos hacer pruebas integradas de nuestro sistema, depurar código, etc. y así poder ajustar los

parámetros de configuración. En este modo Storm simula con threads los distintos nodos del clúster.

8.5. Análisis de datos con Grafos: GraphX Spark, Giraph

GraphX: es el nuevo (alfa) Spark API para gráficos y computación-gráfico paralelo. En un nivel alto, GraphX extiende la Chispa RDD introduciendo el Resilient Distribuido Gráfico propiedad: un multi-grafo dirigido con propiedades asociadas a cada vértice y el borde. Para apoyar gráfico de computación, GraphX expone un conjunto de operadores fundamentales (por ejemplo, subgrafo, joinVertices, y aggregateMessages), así como una variante optimizada de la API Pregel. Además, GraphX incluye una creciente colección de algoritmos de grafos y constructores para simplificar las tareas de análisis gráfico.



8.6. Almacenamiento en pares clave – valor: Redis, Riak, Memcached

Redis.


Redis es un motor de base de datos en memoria, basado en el almacenamiento en hashes (clave, valor) pero también puede ser usada como una base de datos persistente.

Está escrito en ANSI C y es software de código abierto.

Lo más interesante de este tipo de base de datos es su rendimiento que puede muchísimo mayor comparado con motores de bases de datos, y no existe una notable diferencia entre lectura y escritura de datos.

A diferencia de otras bases de datos **clave – valor**, dispone de numerosos tipos de datos complejos como *listas, pilas, colas, conjuntos, conjuntos ordenados y operaciones atómicas definidas para esos tipos de datos.*

Además, es una base de datos en memoria, pero persistente en disco, a diferencia de otros sistemas como memcached que el reinicio de la máquina supone la pérdida de toda la información.



Una de las funcionalidades de redis más desconocidas y más valiosa para aplicaciones en tiempo real es la implementación de Publish/Subscribe pattern para el envío de mensajería. Se trata de una **arquitectura de paso de mensajes desacoplada** (y si se desea distribuida) donde existen remitentes (o publishers) que envían mensajes ante el acaecimiento de un suceso específico sin conocimiento alguno sobre lo que sucede después con el mensaje. Análogamente existen receptores (o subscribers) que será necesario definir a que canales están suscritos, recibiendo todos los mensajes publicados en los canales en que se encuentran.

Características:

- Una característica muy importante sobre el funcionamiento de Redis es que siempre almacena toda la información de la base de datos en memoria, utilizando los discos rígidos como respaldo en caso de cortes de energía. Esta decisión de diseño tiene como consecuencia que la lectura de registros es muy veloz pero la cantidad de registros almacenadas es limitada, dado que no puede exceder el tamaño de la memoria [18]. Esto conlleva a que surja un "patrón de diseño usual" que implica usar un motor como Redis como espacio de caché, haciendo uso eficiente de las estructuras de datos que provee, junto con un motor de base de datos persistente en disco que incluso puede ser relacional [16].
- Redis cuenta con velocidad. Para medir dicha velocidad, incluye una herramienta llamada *redis-benchmark*. Con esta herramienta se puede ver el rendimiento que alcanza en sus operaciones, desde un portátil de la siguiente configuración: CPU: Intel Core i5 520, 2.4 GHz \times 4, Memoria: 3,6 GB, SO Linux Ubuntu 64-bit.
- Se puede ver como el número de peticiones get (obtener el valor de un hash) es de 132k/s y el número de peticiones set (establecer el valor de un hash) es de 135k/s, en lo que queda reflejado lo que anteriormente comentábamos que no existe penalización entre la lectura y escritura de datos.

- Redis crea un único subproceso (single threaded), a diferencia de otros sistemas que crean un proceso o fork por cada petición nueva. En los servidores actuales es muy común tener múltiples núcleos, por lo que se puede dar el caso de desperdiciar toda la potencia del CPU.

En general se debe usar redis cuando los datos no sean críticos y el rendimiento es fundamental. También es recomendable usarlo para guardar sesiones, como caché de las consultas más pesadas de Mysql.

Un ejemplo clarísimo son las aplicaciones en tiempo real como videojuegos, mensajería instantánea, etc.

Riak.

Riak es un motor de bases de datos de código libre escrito en Erlang. Su primera versión fue lanzada por la empresa Basho en 2009 [3] y al momento de redacción del presente trabajo se encuentra en su versión 2.0.2 [26].

Basho destaca para Riak los objetivos de alta disponibilidad, simplicidad, escalabilidad y la carencia de un servidor principal. Por este motivo se propone como una solución ante la necesidad de información que debe estar distribuida en más de cinco servidores debido a su volumen y cuando el tiempo de inactividad es inaceptable. Una particularidad de este motor es su versatilidad al permitir su uso tanto de consistencia eventual como de consistencia fuerte [26]. Este motor utiliza un modelo sencillo de clave-valor para almacenar los datos, permitiendo guardar cualquier tipo de objetos en disco en su representación binaria, desde texto hasta imágenes incluyendo cualquier tipo de documento como JSON, XML y HTML [27] [28].

Redmond y Wilson argumentan que Riak ofrece más posibilidades para la web que ningún otro motor de base de datos, dado que provee una interfaz HTTP REST [17] junto con bibliotecas para las plataformas más utilizadas como Java, Python, Perl, Erlang, Ruby, PHP, .NET, entre otras, junto con soporte integrado un relevamiento de Motores de Bases de Datos No SQL 5 para MapReduce [27]. También incluye funcionalidades más específicas como para navegar entre registros a través de vínculos [3] y soporte de expresiones regulares [4].

Memcached.

Memcached fue desarrollado por Brad Fitzpatrick en 2003, originalmente para LiveJournal [21]. Es de código libre y es considerado como uno de los sistemas No SQL más maduros y de mayor impacto [22].

Los datos que maneja este motor se componen de una clave, un valor, una fecha de expiración y un conjunto de parámetros opcionales. Memcached considera a los tipos como completamente genéricos ya que la información debe ser serializada antes de ser almacenada, argumentando que el servicio de almacenamiento no puede perder tiempo procesando información que puede ser solicitada por cientos o miles de servidores en simultaneo [23].

Memcached, como Redis, también se caracteriza por hacer que la base de datos resida enteramente en memoria, y su función principal es la de optimizar las aplicaciones web aliviando la carga de las bases de datos [21]. Consecuente a esto existe una solución llamada Memcached DB que adopta el protocolo de Memcached, agregándole lógica de persistencia basada en Berkeley DB [5][24].

A pesar de su simplicidad, este motor pertenece a un conjunto que ha inspirado a muchos de los más importantes motores de la actualidad, puesto que demostró que los __índices en memoria pueden ser altamente escalables [3]. Otras fuentes como [25] destacan el rol de este motor en el funcionamiento cotidiano de Internet.

8.7. Almacenamiento en columnas – Bigtable, Cassandra, HBase, Dynamo

Bigtable: es un comprimido , de alto rendimiento, y la propietaria del sistema de almacenamiento de datos construida sobre Google File System , rechoncha de bloqueo del servicio , SSTable (log-estructurado de almacenamiento como LevelDB) y algunos otros de Google tecnologías. El 6 de mayo de 2015, se puso a disposición una versión pública de Bigtable como un servicio. Bigtable también subyace en la nube de Google almacén de datos, que está disponible como parte de la plataforma de Google en la nube.

Cassandra: es un código libre y abierto distribuido sistema de gestión de base de datos diseñada para manejar grandes cantidades de datos a través de muchos servidores de

conveniencia, proporcionando alta disponibilidad con ningún punto único de fallo. Cassandra ofrece soporte robusto para las agrupaciones que abarcan múltiples centros de datos.

HBase: es un código abierto , no relacionales , bases de datos distribuidas siguiendo el modelo de Google BigTable y está escrito en Java . Se desarrolla como parte de Apache Software Foundation 's Apache Hadoop proyecto y se ejecuta en la parte superior de HDFS (Hadoop Distributed File System) , que proporciona capacidades BigTable similar para Hadoop.

Dynamo: es un servicio de base de datos NoSQL completamente administrado que proporciona un desempeño rápido, predecible y muy fácil de escalar. Este servicio permite a los clientes evitar las cargas administrativas que supone tener que utilizar y escalar bases de datos distribuidas, de lo que pasa a ocuparse AWS, de tal modo que no tengan que preocuparse del aprovisionamiento, la instalación y la configuración del hardware, ni tampoco de las tareas de replicación, revisión del software o escalado de clústeres.



8.8. Almacenamiento en motores documentales – MongoDB, CouchDB

Mongodb.

MongoDB es un motor de base de datos de código libre (GPL) escrito en C++ y desarrollado durante 2007, pero su primera versión fue publicada en 2009 [3].

Es la base de datos NoSQL líder y permite a las empresas ser más ágiles y escalables. Es una base de datos ágil que permite a los esquemas cambiar rápidamente cuando las aplicaciones evolucionan, proporcionando siempre la funcionalidad que los desarrolladores esperan de las bases de datos tradicionales, tales como índices secundarios, un lenguaje completo de búsquedas y consistencia estricta.

MongoDB ha sido creado para brindar escalabilidad, rendimiento y gran disponibilidad, escalando de una implantación de servidor único a grandes arquitecturas complejas de centros multidados.

Características.

- ▶ MongoDB brinda un elevado rendimiento, tanto para lectura como para escritura, potenciando la computación en memoria (in-memory). La replicación nativa de MongoDB y la tolerancia a fallos automática ofrece fiabilidad a nivel empresarial y flexibilidad operativa.
- ▶ Organizaciones de todos los tamaños están usando MongoDB para crear nuevos tipos de aplicaciones, mejorar la experiencia del cliente, acelerar el tiempo de comercialización y reducir costes.
- ▶ MongoDB Enterprise ofrece seguridad avanzada, monitorización on-premises, soporte SNMP, certificaciones de SO y mucho más. El servicio de gestión de MongoDB (MMS) ofrece funcionalidad de monitorización y respaldo en la nube o bien on-premises como parte de MongoDB Enterprise.
- ▶ MongoDB provee tres características de diseño: alta eficiencia, alta disponibilidad y fácil escalabilidad [33]. Debido a esta facilidad para manejar proyectos de todo tipo, Redmond lo caracteriza como una herramienta versátil [17].
- ▶ Este motor soporta operaciones atómicas en un documento, y según su documentación oficial, muchas veces esta propiedad es suficiente para resolver los problemas que en un esquema relacional necesitaran de ACID [34]. Esta característica es particularmente señalada por Abramova y Bernardino, junto con la durabilidad de los datos [12]. Otro aspecto también destacado de este sistema es su potente sistema de consultas [5] [17].

CouchDB.

Apache CouchDB es un proyecto escrito en Erlang e iniciado por Damien Katz en abril de 2005, y parte del proyecto Apache desde el año 2008 [3][5]. Uno de sus principios de diseño es que sea una herramienta potente y fácil de usar: el sistema está enfocado de manera tal que sus conceptos principales sean sencillos de entender para quien haya hecho trabajo en la web.

Una de los conceptos que introduce CouchDB es el de “vistas”, que representan la manera principal de acceder a documentos, más allá de las consultas triviales [17].

Estas vistas se definen mediante restricciones a campos y usando funciones JavaScript llamadas *map* y *reduce* que tienen responsabilidades similares al MapReduce de Google [5]. Además, existe un comando para materializar las vistas y guardar los resultados hasta que ocurra una actualización [16].

Con respecto a la escalabilidad, este motor ofrece un enfoque nativamente distribuido al replicar la información de manera asincrónica e incremental, proveyendo consistencia eventual [37]. Además, asume que todo puede fallar en el ambiente distribuido, y es por eso que ofrece durabilidad ante fallas de red y de sistema [3][17].

Los conceptos centrales de CouchDB son simples (pero potentes) y están bien estudiados. Los equipos de operaciones (si tienes un equipo; si no, eres tú) no tienen que temer comportamiento aleatorio o errores que no se pueden encontrar. Si algo sale mal, puedes averiguar fácilmente qué ha pasado-pero estas situaciones no se dan a menudo.

CouchDB también está diseñado para manejar tráfico variable. Por ejemplo, si una página web tiene un pico abrupto de tráfico, CouchDB normalmente absorbe muchas de las consultas concurrentes sin caerse. Probablemente lleve un poco más de tiempo procesar cada consulta, pero todas se responderán. Cuando el pico pasa, CouchDB vuelve a funcionar a su velocidad normal.

La tercera área de relajación se da a la hora de incrementar o reducir el *hardware* que da vida a tu aplicación. Normalmente nos referimos a esto como *escalar* un sistema. CouchDB impone una serie de limitaciones al programador. A primera vista, CouchDB parece inflexible, pero algunas cosas se han dejado fuera conscientemente simplemente porque si CouchDB las tuviera, permitirían al usuario crear aplicaciones que no escalan bien. Exploraremos el tema de escalar CouchDB en la [Parte IV, “Deploying CouchDB”](#).

En resumen: CouchDB no te deja hacer cosas que te podrían meter en un problema más adelante. Esto a veces significa tener que desaprender prácticas y recomendaciones que hayas aprendido en tu trabajo actual o anterior. En el [Capítulo 24, “Recetas”](#) hay una lista de tareas comunes y cómo solventarlas en CouchDB.

Características.

- ▶ CouchDB ha aprendido muchas lecciones de la Web, pero hay una cosa de la Web que se podría mejorar: la latencia. Cuando esperamos a que una aplicación responda o a que una página web se cargue, es casi seguro que estemos esperando a una conexión de red que no es lo rápida que quisiéramos en ese momento. Tener que esperar segundos en vez de milisegundos afecta notablemente a la experiencia de usuario y por ende a la satisfacción del mismo.
- ▶ CouchDB puede solventar el problema de conectividad (cuando no se está conectado), y aquí el tema de escalar vuelve a ser importante. En este caso se trata de escalar hacia abajo. Imagínate CouchDB instalado en teléfonos y otros dispositivos que pueden sincronizar datos con una base de datos central cuando están conectados.

La sincronización no se ve afectada por las limitaciones de una interfaz de usuario requiriendo tiempos de respuesta en rangos de milisegundos. Es más fácil ajustar un sistema para mayor ancho de banda y mayor latencia que para bajo ancho de banda y muy baja latencia. Las aplicaciones móviles pueden usar una instalación local de CouchDB para acceder a datos, y como no hay acceso remoto de red, la latencia por defecto es muy baja.

Couchbase Server.

Couchbase es un motor distribuido de propósito general que provee funcionalidades de caché, clave-valor y almacenamiento de documentos [38]. Este sistema surgió como una unión de la compañía que brindaba soporte comercial para el proyecto Apache CouchDB, CouchOne Inc., junto con los desarrolladores de la empresa Membase Inc. [39]

Couchbase hereda de Membase la funcionalidad para agregar o eliminar servidores elásticamente, moviendo información y redirigiendo pedidos automáticamente en el proceso [3]. Además, ha sido señalado que este motor tiene un mejor manejo de concurrencia que el resto en el rubro [40].

8.9. Almacenamiento en grafos – Neo4j, Titan, OrientDB

Neo4j:

Es una base de datos gráfica nativa altamente escalable que aprovecha las relaciones de datos como entidades de primera clase, para ayudar a las empresas a crear aplicaciones inteligentes para satisfacer los desafíos de datos actuales en constante evolución.

Titan:

Es una graph database bajo licencia Apache 2 escalable y optimizada para almacenar y consultar grafos muy grandes con billones de vertices optimized for storing and querying large graphs with billions of vertices distribuidos en clusters de muchas máquinas.

OrientDB:

Es un código abierto NoSQL sistema de gestión de base de datos escrito en Java. Es una base de datos multi-modelo, el apoyo gráfico, documento, clave / valor, y el objeto modelos, pero las relaciones se gestionan como en bases de datos de gráficos con conexiones directas entre los registros. Es compatible con los modos de esquema mezclado sin esquema, el esquema completo y. Cuenta con un sistema de perfiles de seguridad fuerte basada en usuarios y roles y apoyos consultar con Gremlin junto con SQL extendido por recorrido del grafo.


8.10.Computación distribuida con contenedores – Docker, Mesos

Docker

Es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de Virtualización a nivel de sistema operativo en Linux.

Con la tecnología de Docker podremos virtualizar un Linux con todas las aplicaciones que necesitemos dentro de nuestro sistema operativo Linux, para "empaquetarlo" y desplegarlo en cualquier otro Linux sin necesidad más que de introducir un par de comandos.

Portabilidad.



El contenedor Docker podremos desplegarlo en cualquier otro sistema (que soporte esta tecnología), con lo que nos ahorraremos el tener que instalar en este nuevo entorno todas aquellas aplicaciones que normalmente usamos.

Ligereza.

El peso de este sistema no tiene comparación con cualquier otro sistema de virtualización más convencional que estemos acostumbrados a usar. Por poner un ejemplo, una de las herramientas de virtualización más extendida es VirtualBox, y cualquier imagen de Ubuntu que queramos usar en otro equipo pesará entorno a 1Gb si contamos únicamente con la instalación limpia del sistema. En cambio, un Ubuntu con Apache y una aplicación web, pesa alrededor de 180Mb, lo que nos demuestra un significativo ahorro a la hora de almacenar diversos contenedores que podamos desplegar con posterioridad.

Autosuficiencia.

Un contenedor Docker no contiene todo un sistema completo, sino únicamente aquellas librerías, archivos y configuraciones necesarias para desplegar las funcionalidades que contenga. Asimismo, Docker se encarga de la gestión del contenedor y de las aplicaciones que contenga.

Además, su ligereza hará las delicias del usuario, puesto que incluso en equipos con algunos años a sus espaldas se desenvuelve prácticamente igual que el sistema anfitrión, aparte de ofrecernos un entorno similar a Git para, a base de "capas", controlar cada cambio que se haga en la máquina virtual o contenedor.

¿Cómo funciona Docker?

En un principio contamos con una **imagen base**, sobre la que realizaremos los diferentes cambios. Tras **confirmar estos cambios mediante la aplicación Docker**, crearemos la imagen que usaremos. Esta imagen contiene únicamente las diferencias que hemos añadido con respecto a la base. Cada vez que queramos ejecutar esta imagen necesitaremos la base y las 'capas' de la

imagen. Docker se encargará de **acoplar la base, la imagen y las diferentes capas con los cambios** para darnos el entorno que queremos desplegar para empezar a trabajar.

Desde su sitio oficial podemos ver que está disponible para **Ubuntu, ArchLinux, Gentoo, Fedora, OpenSUSE y FrugalWare**, así como desde el propio código binario de la aplicación. Además también tenemos indicaciones para poder desplegar contenedores bajo entornos **Windows, Mac, Amazon EC2, Rackspace o Google Cloud**.

principales características y funcionalidades de Docker:

Autogestión de los contenedores.

Fiabilidad.

- ▶ Aplicaciones **libres** de las **dependencias** instaladas en el sistema anfitrión.
- ▶ Capacidad para **desplegar multitud de contenedores** en un mismo equipo físico.
- ▶ Puesta en marcha de los servicios en un abrir y cerrar de ojos.
- ▶ Contenedores muy **livianos** que facilitan su almacenaje, transporte y despliegue.
- ▶ Capacidad para ejecutar una **amplia gama de aplicaciones** (prácticamente cualquier cosa que se nos ocurra podrá ejecutarse en un contenedor Docker).
- ▶ Compatibilidad **Multi-Sistema**, podremos desplegar nuestros contenedores en multitud de plataformas.
- ▶ La aplicación base de Docker **gestionará los recursos existentes** para asignarlos responsablemente entre los contenedores desplegados.
- ▶ Podremos establecer una base desde la que comenzar nuestros proyectos, lo que nos ahorrará el tiempo de preparar el entorno para cada uno de ellos.

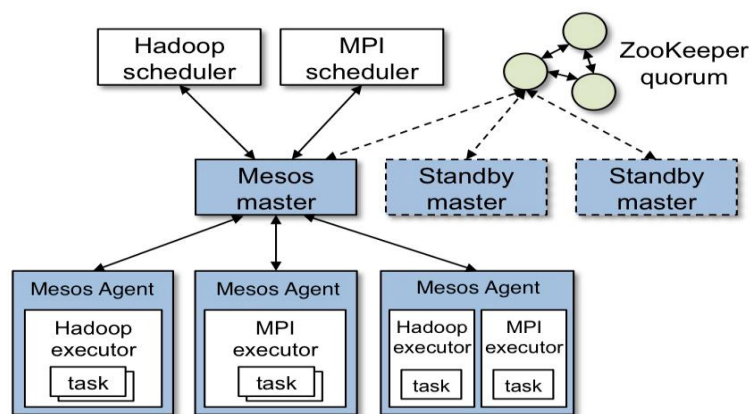
Apache mesos

Mesos nos ofrece una capa de abstracción entre los servidores y los recursos, es un concepto un poco diferente, pero a mi parecer cuando lo entiendes me parece excepcional. Por otro lado Mesos te proporciona una gestión de cluster y como una gestión de los recursos del cluster.

¿Qué ofrece Mesos?

- ▶ Escalabilidad hasta 10000 nodos.
- ▶ Alta disponibilidad de los servidores Master y Slaves a través de Zookeeper.
- ▶ Soporta de forma nativa Docker.
- ▶ Aislamiento de recursos entre procesos en el clúster, o sea que aseguramos que no se pisen entre ellos o que no puedan acceder los unos a los otros.
- ▶ Podemos desarrollar Frameworks en Java, Python y C++.

Arquitectura de Mesos.



Descripción de la figura.

1. Agente de 1 informes a la maestra que tiene 4 CPU y libre de 4 GB de memoria. El maestro entonces invoca el módulo de directivas de asignación, que le dice que el marco 1 se debe ofrecer todos los recursos disponibles.
2. El maestro envía una oferta de recursos que describe lo que está disponible en el agente 1 a 1 marco.
3. planificador del marco responde a la principal con información sobre dos tareas a ejecutar en el agente, usando <2 CPU, 1 GB de RAM> para la primera tarea, y <1 CPU, RAM de 2 GB> para la segunda tarea.

4. Por último, el maestro envía a las tareas del agente, que asigna los recursos apropiados para ejecutor del marco, que a su vez pone en marcha las dos tareas (representado con las fronteras con líneas de puntos en la figura). Debido a 1 CPU y 1 GB de RAM son todavía no asignado, el módulo de asignación puede ahora les ofrecen a marco 2.

Podremos **compartir** nuestros contenedores para aumentar los repositorios de Docker así como beneficiarnos de los que compartan los demás.

8.11.Programación Reactiva: Scala(Akka), Java Rx

Scala(Akka): Es La manipulación de grandes volúmenes de datos distribuidas en un clúster utilizando conceptos funcionales está muy extendido en la industria, y es sin duda uno de los primeros usos industriales de las ideas funcionales. Esto se evidencia por la popularidad de MapReduce y Hadoop, y más recientemente Spark Apache, una, en memoria distribuida marco de las colecciones rápido escrito en Scala.

RxJava: es una librería ideal para usar en Android, ya que para conseguir una buena experiencia de usuario recomiendan hacer todas las operaciones en un hilo en background, lo que obliga a todos los desarrolladores de Android a lidiar con problemas de concurrencia y asincronía.

8.12.Arquitectura Web de Micro servicios: NodeJs, Dropwizard, Spring Boot,etc. Node.js

Node.js es un entorno Javascript del lado del servidor, basado en eventos. Node ejecuta javascript utilizando el motor V8, desarrollado por Google para uso de su navegador Chrome. Aprovechando el motor V8 permite a Node proporciona un entorno de ejecución del lado del servidor que compila y ejecuta javascript a velocidades **increíbles**. El aumento de velocidad es importante debido a que V8 compila Javascript en código de máquina nativo, en lugar de interpretarlo o ejecutarlo como bytecode. Node es de código abierto, y se ejecuta en Mac OS X, Windows y Linux.

Pero ¿por qué javascript del lado del servidor?

Aunque Javascript tradicionalmente ha sido relegado a realizar tareas menores en el navegador, es actualmente un lenguaje de programación totalmente, tan capaz como cualquier otro lenguaje tradicional como C++, Ruby o Java. Además Javascript tiene la ventaja de poseer un excelente modelo de eventos, ideal para la programación asíncrona. Javascript también es un lenguaje omnipresente, conocido por millones de desarrolladores. Esto reduce la curva de aprendizaje de Node.js, ya que la mayoría de los desarrolladores no tendrán que aprender un nuevo lenguaje para empezar a construir aplicaciones usando Node.js.

Además de la alta velocidad de ejecución de Javascript, la verdadera magia detrás de Node.js es algo que se llama Bucle de Eventos (Event Loop). Para escalar grandes volúmenes de clientes, todas las operaciones intensivas I/O en Node.js se llevan a cabo de forma asíncrona. El enfoque tradicional para generar código asíncrono es engorroso y crea un espacio en memoria no trivial para un gran número de clientes (cada cliente genera un hilo, y el uso de memoria de cada uno se suma). Para evitar esta ineficiencia, así como la dificultad conocida de las aplicaciones basadas en hilos, (programming threaded applications), Node.js mantiene un event loop que gestiona todas las operaciones asíncronas.

Dropwizard

Es una combinación pertinaz de varias herramientas y marcos Java ligeros, muchos de los cuales merecen una mención en su propio derecho. El paquete incluye gran número de nuestras técnicas favoritas, incluyendo un servidor HTTP integrado, soporte para puntos finales REST, una función de las métricas operacionales y controles de salud, y las implementaciones sencillas. Dropwizard hace que sea fácil de hacer lo correcto, lo que le permite concentrarse en la complejidad esencial de un problema en lugar de la tubería.

Spring Boot.

Es un sub-proyecto de Spring, el mismo busca facilitarnos la creación de proyectos con framework Spring eliminando la necesidad de crear largos archivos de configuración xml, Spring

Boot provee configuraciones por defecto para Spring y otra gran cantidad de librerías, además provee un modelo de programación parecido a las aplicaciones java tradicionales que se inician en el método main.

8.13.HTML5: Componentes Web, Frameworks, Web Sockets.

WebSockets:

Hacer una app real-time (en tiempo real) es una gran mejora y los WebSockets brindan esta habilidad para agregar comunicación bi-direccional en una conexión persistente para tu aplicación, permitiendo incrementar la interactividad y el compromiso con el usuario.

WebSockets proporcionan una conexión permanente entre un cliente y servidor que ambas partes pueden utilizar para empezar a enviar datos en cualquier momento. El cliente establece una conexión WebSocket a través de un proceso conocido como el protocolo de enlace WebSocket. Este proceso comienza con el cliente envía una solicitud HTTP regular para el servidor. Una Upgrade cabecera se incluye en esta solicitud que informa al servidor que el cliente desea establecer una conexión WebSocket.

Frameworks:

Es un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación. Sí, es una definición muy genérica, pero también puede serlo un framework: sin ir más lejos, el paradigma MVC (Model-View-Controller) dice poco más que “separa en tu aplicación la gestión de los datos, las operaciones, y la presentación”. En el otro extremo, otros frameworks pueden llegar al detalle de definir los nombres de ficheros, su estructura, las convenciones de programación, etc

Los frameworks no necesariamente están ligados a un lenguaje concreto, aunque sea así en muchas ocasiones. En el cada vez más popular Ruby on Rails, ‘Ruby’ es el lenguaje de programación y ‘Rails’ el framework; por otro lado, JavaServer Faces está orientado a desarrollos en Java. Sin embargo, nada impide definir el mismo framework para lenguajes diferentes: por ejemplo, existe un framework llamado Biscuit cuyo objetivo es prácticamente convertirse en un “PHP on Rails”. Eso sí, cuanto más detallado es el framework, más necesidad

tendrá de ceñirse a un lenguaje concreto. También es posible que el framework defina una estructura para una aplicación completa, o bien sólo se centre en un aspecto de ella.

Los Web Components: nos ofrecen un estándar que va enfocado a la creación de todo tipo de componentes utilizables en una página web, para realizar interfaces de usuario y elementos que nos permitan presentar información (o sea, son tecnologías que se desarrollan en el lado del cliente). Los propios desarrolladores serán los que puedan, en base a las herramientas que incluye Web Components crear esos nuevos elementos y publicarlos para que otras personas también los puedan usar.

Ejemplos clásicos de Web Components


El ejemplo más típico que veremos por ahí es un mapa de Google. Hoy, si no usamos web components, cuando queremos mostrar un mapa en una página web, tenemos que crear código en tres bloques.

- 1.3. Un HTML con el elemento donde se va a renderizar el mapa
- 1.4. Un CSS para definir algún estilo sobre el mapa, por ejemplo sus dimensiones

Lo más importante, un Javascript para que puedas generar el mapa, indicando las coordenadas que desees visualizar (para centrar la vista inicial) y muchos otros detalles de configuración que tu mapa necesite.

9. Conclusiones

La investigación sobre las diferentes tecnologías disruptivas nos ha dejado un importante enriquecimiento intelectual, ya que son tecnologías que aportan una dinámica fundamental a la arquitectura del software. Pueden considerarse, además, como las innovaciones que hacen los productos y servicios más accesibles y asequibles. Con lo cual, se llega con la tecnología a un mayor grupo de la comunidad tecnológica.



Algunas de las tecnologías analizadas en este trabajo son de distribución libre y pueden requerir poca o ninguna inversión de capital. La tecnología disruptiva podría ayudar a nivelar el escenario tecnológico, aportando el diseño, producción y distribución de productos y servicios al alcance de los usuarios. Cada una de estas tecnologías podrá tener un gran potencial para impulsar el crecimiento económico e incluso cambiar las fuentes de ventajas comparativas entre las naciones.

Otro factor importante del presente trabajo se relaciona con el proyecto de grado de la Maestría, ya que se aplicarán algunas de las tecnologías investigadas y aprendidas en el desarrollo del mismo proyecto.

10. Referencias Bibliográficas

<https://sg.com.mx/revista/50/un-vistazo-apache-spark-streaming#.WBQoofp97IV>

<https://www.adictosaltrabajo.com/tutoriales/introduccion-storm/#02>

<https://msandovalcris.wordpress.com/>

<http://www.antweb.es/servidores/redis-todo-lo-que-debes-saber>

<https://www.mongodb.com/es>

<http://guide.couchdb.org/editions/1/es/why.html>

3. Cattell, Rick: Scalable SQL and NoSQL Data Stores. SIGMOD Rec., 39(4):12{27, Mayo 2011, ISSN 0163-5808. <http://doi.acm.org/10.1145/1978915.1978919>.

4. Hecht, Robin y Jablonski, Stefan: NoSQL Evaluation: A Use Case Oriented Survey. En Proceedings of the 2011 International Conference on Cloud and Service Computing, CSC '11, p_aginas 336{341, Washington, DC, USA, 2011. IEEE Computer Society, ISBN 978-1-4577-1635-5. <http://dx.doi.org/10.1109/CSC.2011.6138544>.

Un Relevamiento de Motores de Bases de Datos NoSQL

Strauch, Christof: NoSQL Databases. Lecture: Selected Topics on Software-Technology Ultra-Large Scale Sites, Manuscript, Hochschule der Medien, Stuttgart (Stuttgart Media University), 2011.

17. Redmond, Eric y Wilson, Jim R.: Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement. Pragmatic Bookshelf, 2012, ISBN 1934356921, 9781934356920.

Howe, Bill: Introduction to Data Science. Online Course, Septiembre 2014. <https://www.coursera.org/course/datasci>.

21. memcached: About Memcached. <http://www.memcached.org/about>, visitado el 2010-09-30.

22. Howe, Bill: Introduction to Data Science. Online Course, Septiembre 2014. <https://www.coursera.org/course/datasci>.

23. memcached wiki: New overview, Agosto 2011. <https://code.google.com/p/>

memcached/wiki/NewOverview, visitado el 2010-09-30.

24. MemcacheDB, Enero 2009. <http://memcachedb.org>, visitado el 2010-09-30.

25. Aylward, Lee: Gimme the cache! memcached turns 10 years old, Mayo 2013. <http://arstechnica.com/information-technology/2013/05/gimme-the-cache-memcached-turns-10-years-old-today>.

26. Basho Technologies: Riak: Riak Docs: Why Riak? <http://docs.basho.com/riak/latest/theory/why-riak>, visitado el 2010-09-30.

27. Basho Technologies: Riak. <http://basho.com/riak>, visitado el 2010-09-30.

28. Basho Technologies: Riak: From Relational to Riak { A Technical Brief. <http://basho.com/assets/RelationaltoRiak.pdf>, visitado el 2010-09-30.

33. MongoDB: Introduction to MongoDB. <http://docs.mongodb.org/manual/core/introduction>, visitado el 2010-09-30.

37. Anderson, J. Chris, Lehnardt, Jan y Slater, Noah: CouchDB: The Definitive Guide { Time to Relax. O'Reilly Media, Inc., 1ra edici_on, 2010, ISBN 0596155891, 9780596155896.

38. Couchbase Server: About. <http://www.couchbase.com/nosql-databases/about-couchbase-server>, visitado el 2010-09-30.

39. Couchbase Server: Couchbase vs. Apache CouchDB { A comparison of two open source NoSQL database technologies. <http://www.couchbase.com/couchbase-vs-couchdb>, visitado el 2010-09-30.

40. Oliver, Andrew: Couchbase 2.0: This means war, Diciembre 2012. <http://www.infoworld.com/article/2616396/application-development/couchbase-2-0--this-means-war.html>.

41. Couchbase Server: Customers. <http://www.couchbase.com/case-studies>, visitado el 2010-09-30.