stratebi
open business intelligence

Benchmark Big Data OLAP Analytics

Apache Kylin™ VS VERTICA

PostgreSQL

1. INTRODUCCIÓN

El usuario de sistemas Business Intelligence (BI) está acostumbrado a una respuesta muy rápida, interactiva, cuando hace uso de cuadros de mando, informes y consultas analíticas de detalle. Las aplicaciones BI que cumplen este requisito de procesamiento interactivo se conocen como aplicaciones OLAP (On-Line Analytical Processing).

Sin embargo, cuando trabajamos con fuentes de datos con características Big Data (Volumen, Variedad y Velocidad), nuestras tablas de métricas (ej. volumen de ventas, unidades…) y las que describen el contexto (ej. fecha, cliente, producto) pueden llegar a almacenar miles de millones de filas, haciendo que los requisitos de procesamiento sean realmente exigentes incluso para las tecnologías Big Data más actuales.

Con el objetivo de soportar las aplicaciones OLAP con Big Data, en los últimos años han aparecido múltiples tecnologías que prometen excelentes resultados. Algunas de las más conocidas son Apache Kylin, Vertica, Druid, Google Big Query o Amazon Red Shift.

Aunque las arquitecturas que proponen son bien distintas, todas ellas prometen la posibilidad de realizar consultas de tipo analítico con tiempos de respuesta (latencia de consulta) que van de milisegundos a unos pocos segundos y usando el lenguaje estándar de consulta SQL, bien conocido por la mayoría de equipos de IT de una organización.

Sin embargo, nuestra experiencia en la implantación de estas tecnologías nos ha demostrado que no todas cumplen los resultados que prometen. Apache Kylin y Vertica son las tecnologías con las que mejores resultados hemos logrado hasta el momento en nuestras implantaciones de sistemas Big Data OLAP. Con el objetivo de poner a prueba y comparar el rendimiento de estas tecnologías y de estas frente a otras tecnologías no Big Data, como PostgreSQL, surge la idea de llevar a cabo el benchmark Big Data OLAP que presentamos en este artículo.

2. TECNOLOGÍAS BIG DATA OLAP

A continuación, describiremos las tecnologías Big Data OLAP que forman parte de la comparativa: Apache Kylin y Vertica. Además de comparar entre sí estás tecnologías, también las hemos comparado con la base de datos relacional PostgreSQL. Esta tecnología open source, a pesar de no ser Big Data, suele ofrecer muy buenos resultados con aplicaciones OLAP, por lo que nos ha parecido interesante incluirla en el benchmark para poder medir las diferencias con Kylin y Vertica en un escenario Big Data OLAP.

Apache Kylin

Herramienta open source que se define en su web como un motor distribuido para el procesamiento analítico, proporciona una interfaz SQL y soporta aplicaciones de análisis multidimensional (OLAP) sobre una plataforma Hadoop/Spark y fuentes de datos de tipo Big Data. Originalmente desarrollada en eBay por Luke Han y otros investigadores, a finales de 2014 fue liberada como proyecto Apache open source y su desarrollo ha continuado de forma incesante hasta nuestros días, como así demuestra la activad del proyecto.

A continuación, enumeramos las características principales de Kylin:

- **Consultas interactivas, con latencias inferiores al segundo, sobre modelos analíticos con tablas de más de 10 mil millones de filas**.
- **Soporte para el lenguaje de consulta estándar SQL.**
- **Compresión avanzada de los datos.**
- **Drivers J/OBDC para la integración con las herramientas de BI más populares:** Power BI, Tableau, Pentaho, Zeppelin, Superset , Microstrategy, …
- **Escalabilidad total** añadiendo máquinas al clúster Hadoop y con soporte para el almacenamiento en HDFS y S3. También soporta la **alta disponibilidad.**
- **Completas interfaz web y API restful,** para gestionar los modelos de datos, modelos de cubos, cargas y actualizaciones de cubos, configuración e incluso la consulta de datos y la recuperación de resultados en formato JSON.
- **Optimizador de cubos automático** en base a la recopilación de estadísticas y la aplicación de técnicas de Machine Learning.

- **Seguridad** mediante la integración con sistemas LDAP y Apache Ranger, para la seguridad a nivel de funcionalidades, esquemas de cubos y datos.
- **Versión comercial con características añadidas:** Además de la versión open source que hemos probado en esta comparativa, existe una versión comercial denominada Kyligence. Esta versión añade algunas características mejoradas como un sistema de almacenamiento de mayor rendimiento, asistentes inteligentes para el diseño de los cubos, seguridad mejorada y soporte técnico.

En la Figura 1. se muestra la arquitectura del sistema Apache Kylin. El almacén de datos (Data Warehouse, DW), que usamos como fuente de datos del sistema Kylin, es un conjunto de tablas que se mantiene en Apache Hive u otros sistemas de bases de datos (ej. SQL Server, MySQL o incluso Vertica). De forma adicional también permite la entrada de datos en tiempo real desde una cola (topic o tabla streaming) de Apache Kafka.

Una vez seleccionada la fuente de datos, es necesario definir el modelo de datos desde la interfaz web de Apache Kylin, dónde se indican aspectos cómo las uniones (joins) entre las distintas tablas, las columnas a usar como métricas o dimensiones y las columnas de fecha usadas para la carga incremental y actualización de datos.

A continuación, el último paso es la definición del cubo OLAP, indicando ciertos aspectos relacionados con nivel de pre agregación y pre combinación de los datos. Este tipo de arquitecturas OLAP basadas en la pre agregación y pre combinación de los datos se denominan Multidimensional OLAP (M-OLAP) y, gracias a esta característica, Kylin logra unos excelentes tiempos de consulta.

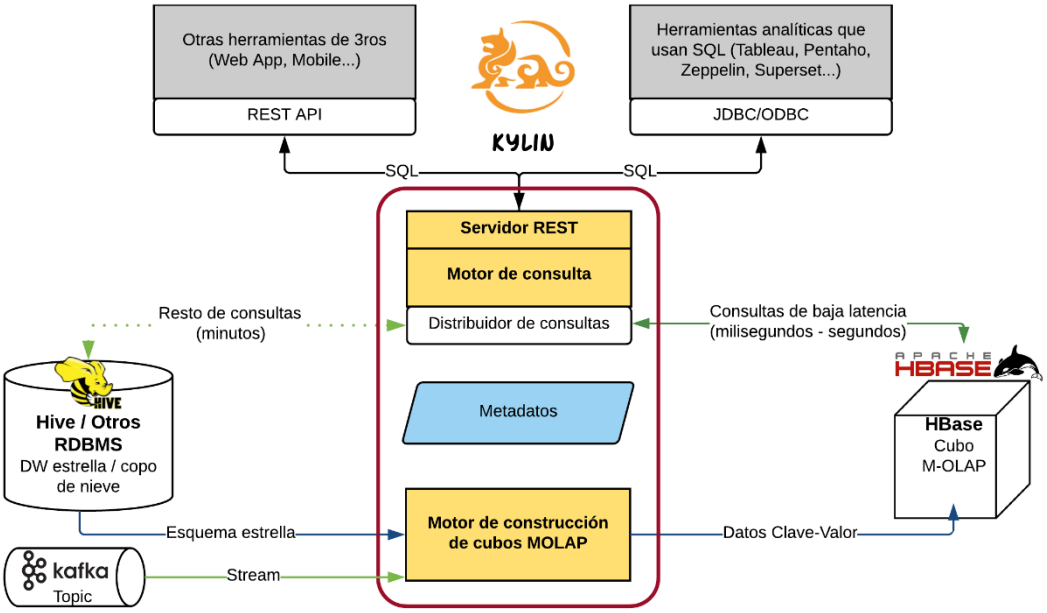


Figura 1. Arquitectura del sistema Apache Kylin.

Una vez se he hecho una primera carga de datos (construcción del cubo), ya es posible ejecutar consultas SQL o iniciar nuevos procesos de construcción para añadir o recargar datos del cubo. Las consultas sobre el cubo definido en Kylin se resuelven en menos de 1 segundo en la mayoría de los casos.

Más de 200 empresas ya usan Kylin a nivel mundial, entre ellas StrateBI como así se recoge en la lista de empresas que usan y apoyan el proyecto Kylin.

Vertica

Vertica es una base de datos analítica con una arquitectura de procesamiento de datos masivo en paralelo (MPP), almacenamiento columnar y, al igual que Kylin, orientada a dar soporte a aplicaciones OLAP sobre fuentes de datos Big Data. Sus inicios se remontan al año 2005, cuando el investigador Michael Sonebraker público el paper "C-Store: A Column-oriented DBMS" que es la base del sistema Vertica. Posteriormente fue adquirido por Hewlett Packard (HP) y más tarde por Microfocus, en 2017.

A diferencia de Kylin, Vertica es una tecnología que puede funcionar como un clúster independiente de las tecnologías Hadoop. A continuación, enumeramos las características principales de Vertica:

- **Consultas interactivas sobre tablas de miles de millones de filas**.
- **Soporte para el lenguaje de consulta estándar SQL.**
- **Compresión avanzada de los datos.**
- **Drivers J/OBDC para la integración con las herramientas de BI más populares:** Power BI, Tableau, Microstrategy, Pentaho…
- **Escalabilidad total** añadiendo máquinas al clúster de Vertica y **alta disponibilidad**.
- **Completa interfaz web,** para gestionar el estado y carga del clúster (general y detalle por nodo), las bases de datos, esquemas, tablas, el asistente para la optimización de esquemas de forma automática o el análisis del plan de consulta.
- **Optimizador de esquemas semi-automático** mediante un sencillo asistente.
- **Seguridad** con autenticación y autorización de usuario, seguridad a nivel de esquema y tablas, integración con LDAP y encriptación de las comunicaciones cliente-servidor.
- **Funciones para Machine Learning nativas:** Incluye funciones nativas de preparación de datos, algoritmos de regresión, clasificación y clustering.
- **Funciones de usuario (UDF):** Característica muy útil que nos permitirá crear funciones para usar en nuestras consultas SQL con lenguajes como SQL, Python, C++, Java o R.
- **Versión comercial con características añadidas:** La versión gratuita está limitada en procesamiento a un clúster de 3 máquinas y en almacenamiento a 1 Tb. La versión comercial no tiene ningún tipo de limitación en cuanto a la escalabilidad.

En la Figura 2. se muestra la arquitectura del sistema Vertica. Para crear un cubo OLAP en Vertica es necesario definir un esquema de tablas y cargarlo desde alguna de las fuentes de datos soportadas: ficheros, bases datos relacionales, mediante herramientas de ETL como Pentaho o Talend, o incluso desde colas (topics) de Kafka, para la carga de datos en tiempo real.

Tras esto, ya podemos ejecutar consultas en lenguaje SQL o conectar desde alguna de las herramientas de BI más conocidas, como Tableau, Power BI o Pentaho, para crear nuevos cuadros de mando e informes con respuesta interactiva.

A diferencia de Kylin, Vertica no pre-agrega y pre-combina los datos, por lo que la resolución de las consultas analíticas requiere más computación en tiempo de consulta.

Sin embargo, podemos ejecutar el asistente de optimización para aplicar los mecanismos de proyecciones y flattened tables, los cuales permiten lograr cierto nivel de pre-agregación y pre-combinación, para la mejora del tiempo de respuesta de las consultas. Este tipo de aproximaciones híbridas son denominadas Hybrid OLAP (H-OLAP).

Figura 2. Arquitectura del sistema Vertica.

Además, el clúster de Vertica es capaz de integrarse con un clúster Hadoop, para hacer consultas sobre los datos almacenados en Hive o HDFS, combinarlos con los de las tablas de Vertica para enriquecer el análisis o aprovechar las ventajas de herramientas como Spark para el procesamiento de Big Data menos estructurado.

En la actualidad Vertica se ha implantado en un gran número de empresas con éxito. En StrateBI apostamos por esta tecnología, logrando su implantación con éxito en múltiples sectores y siendo partners de Vertica.

3. BENCHMARK

Una vez presentadas las tecnologías de esta comparativa de rendimiento, en las siguientes secciones describiremos los detalles del benchmark implementado y la configuración de las pruebas realizadas.

Implementación del benchmark

En la actualidad existen múltiples benchmark de bases de datos, siendo el benchmark TPC uno de los más conocidos y aceptados por los líderes de la industria como Oracle o Microsoft. En nuestro caso hemos usado el benchmark SSB, una variante del TPC-H , que propone un esquema "en estrella" más optimizado para sistemas OLAP que el propuesto originalmente por el **benchmark TPC-H**.

En la Figura 3. se muestra dicho esquema, el cual describe un escenario típico del sector retail, donde las ventas de productos (cantidad, precio, con/sin descuento e impuestos…) se estudian a un nivel de detalle de línea de venta por fecha (año, mes y día), producto, cliente y proveedor.

El benchmark también selecciona y adapta 13 de las consultas SQL que propone TPC-H para el benchmark. Se trata de 13 consultas en lenguaje SQL (con cláusulas group by, where, joins y funciones de agregación) cuyo diseño está fundamentado científicamente para la prueba y se recoge en el Anexo 1 de este documento.

Además, para su implementación, el benchmark requiere adaptar el generador de datos que proporciona TPC-H. Para ello hemos usado la implementación del equipo de Kyligence, aplicando algunas modificaciones en el código para su uso con Kylin, Vertica y PostgreSQL.

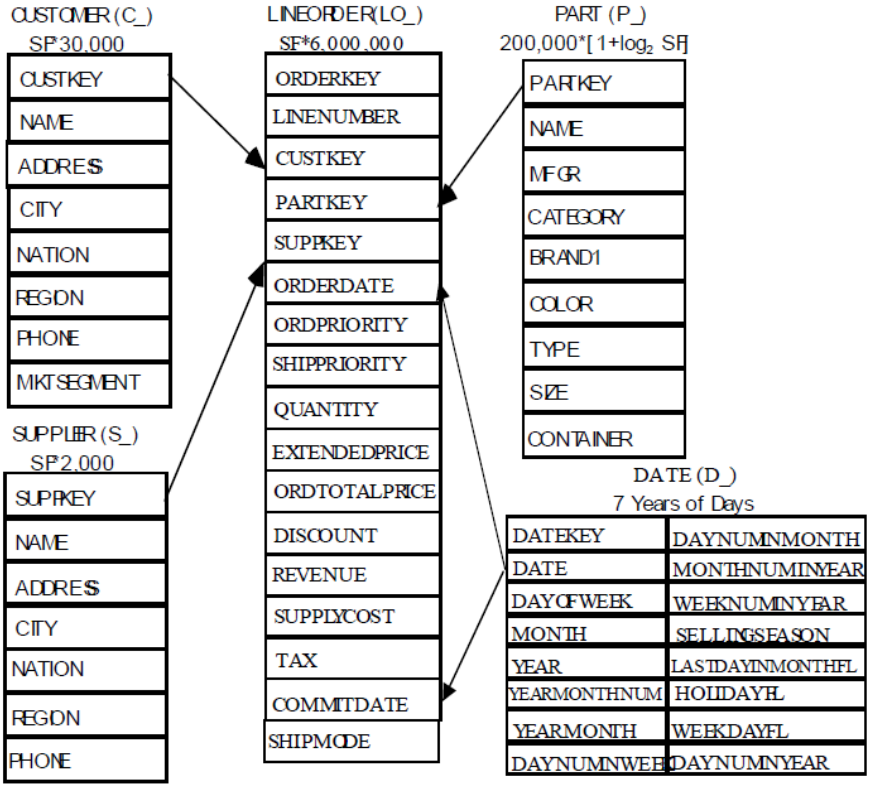


Figura 3. Esquema en estrella de un escenario del sector retail propuesto por el benchmark SSB

Configuración de las pruebas realizadas

La configuración de las pruebas realizadas tiene tres aspectos clave, enumerados a continuación en el orden en que fueron determinados:

- **Volumen de filas de las tablas** de hechos (LINE ORDER) **a cargar** y de las tablas de dimensión (DATE, CUSTOMER, SUPPLIER, PART).
- **Hardware** seleccionado.
- **Optimizaciones** propias de cada sistema sobre el modelo de datos.

Para la prueba se ha configurado un escenario típico de Big Data OLAP con una **tabla de hechos (LINE_ORDER) muy grande y tablas de dimensión (CUSTOMER, PART, SUPPLIER y DATE) de tamaño grande (cientos de miles de filas)**.

El **volumen de filas a cargar** en las tablas es un aspecto que puede configurarse indicando los valores de filas para cada tabla en un archivo de configuración del generador de datos del SSB y, además, incrementarlo indicando un factor de escalado al ejecutar dicho generador.

El tamaño de filas de las tablas se ha configurado de forma que en las pruebas más sencillas represente un reto para sistemas OLAP sobre tecnología relacional tradicional, como PostgreSQL.

En las pruebas más complejas, el número de filas es lo suficientemente alto como para suponer un reto incluso para los sistemas Big Data OLAP actuales. En la Tabla 1. se muestra la configuración en número de filas por tabla para cada una de las tres pruebas realizadas.

	LINEORDER	CUSTOMER	PART	SUPPLIER	DATE
Prueba - Tipo de Tabla	Hechos (KPI)	Dimensión	Dimensión	Dimensión	Dimensión
100M	100.000.000	40.000	32.000	20.000	2.556
500M	500.000.000	200.000	48.000	100.000	2.556
1.000M	1.000.000.000	400.000	56.000	200.000	2.556

Tabla 1. Configuración en volumen de filas de cada prueba realizada.

Para implementar estás pruebas con los sistemas OLAP de la comparativa, se ha elegido el **hardware y versiones** de las herramientas cuyo detalle se muestra en la Tabla 2.

Tecnología	Procesamiento Distribuido	Tipo de Instancias	Nº Instancias	Procesador	Cores	Memoria RAM
Kylin 2.4	Sí	Nube dedicada	3	Intel(R) Atom(TM) CPU C2750 @ 2.40GHz	8	32 Gb
Vertica 9.1	Sí	Nube dedicada	3	Intel(R) Atom(TM) CPU C2750 @ 2.40GHz	8	32 Gb

PostgreSQL 9.6	No	Nube dedicada	1	Intel(R) Atom(TM) CPU C2750 @ 2.40GHz	8	32 Gb

Tabla 2. Hardware y versiones de las herramientas empleados en el benchmark.

Para Kylin y Vertica el hardware seleccionado es el mismo. Sin embargo, en el caso de PostgreSQL solo se ha usado una instancia, pues se trata de un sistema no distribuido que no es posible montar en clúster.

En el caso Kylin y Vertica es importante remarcar que mientras que Vertica funciona en un clúster exclusivo, Kylin se ha instalado en una de las máquinas de un clúster Hadoop con distribución Hortonworks, compartiendo por tanto recursos con este. En la práctica es habitual instalar Kylin en una máquina separada del clúster Hadoop o, incluso, en modo clúster (clúster de Kylin) para mejorar el rendimiento en consulta mediante el balanceo de carga.

Por último, es importante indicar que en la implementación del modelo de datos hemos aplicado algunas **técnicas u optimizaciones básicas de cada sistema**. Solo hemos aplicado las más comunes, aquellas que se suelen usar en cualquier implementación, para tratar de realizar una comparativa en igualdad de condiciones entre las 3 tecnologías:

- Apache Kylin
  - Definición de las columnas de las tablas de dimensión como "Normales" para que se incluyan en la pre-agregación (M-OLAP) de los datos del cubo resultante. Las consultas que implican columnas de dimensión "Normales" no requerirán apenas computación en tiempo consulta.

- Vertica
  - Aplicación del optimizador de modelos usando las consultas del benchmark como referencia. El resultado es la optimización automática del modelo mediante la generación de proyecciones. Las proyecciones son sub conjuntos del modelo optimizadas (datos ordenados, codificación más adecuada, ...) para dar respuesta optimizada a los tipos de consultas analíticas más frecuentes en ese sistema OLAP.

- PostgreSQL.
    - Definición de índices en columnas de unión en entre la tabla de hechos (Foreign Keys) y las tablas de dimensión (Surrogate Keys). Es una técnica habitual en la implementación de modelos OLAP con bases de datos relacionales.

Resultados del benchmark y análisis

Una vez generados los datos, hemos ejecutado todas y cada una de las consultas analíticas en lenguaje SQL propuestas por el benchmark SSB.

Para obtener las muestras de latencia de cada consulta, hemos usado las herramientas de medición de tiempos que proporcionan Kylin, Vertica y PostgreSQL, de forma que la medición sea lo más precisa posible. Además, el resultado de cada consulta se obtiene a partir del promedio de 20 muestras. En el caso de Kylin se ha desactivado la caché de consulta que incorpora para evitar el beneficio adicional que supondría su uso.

En la Tabla 3. mostramos los resultados de las pruebas. Para analizar mejor los resultados hemos usado la siguiente codificación de color:

- Verde: Consultas con tiempo de respuesta o latencia inferior a 5 segundos. Es un tiempo adecuado para permitir la consulta interactiva (OLAP) de los datos.
- Azul: Latencia < 10 segundos. El usuario es consciente de un pequeño retraso en estas consultas, aunque en muchos casos se considera un tiempo aceptable.
- Naranja: Latencia < 20 segundos. La interactividad se ve afectada por los tiempos de consulta y en muchos escenarios OLAP podrían considerarse inaceptables.
- Granate: Latencia >= 20. Estos tiempos de consulta son inaceptables para un sistema de tipo OLAP, ya que no permiten mantener la interactividad.

Prueba	P1 – 100M (segundos)			P1 – 500M (segundos)			P1 – 1.000M (segundos)		
Consulta	Kylin	Vertica	Postgre	Kylin	Vertica	Postgre	Kylin	Vertica	Postgre
Q1.1	0.2	0.2	22.4	0.3	0.3	+280	0.6	0.6	-
Q1.2	0.2	0.4	18.7	0.3	0.2	+280	0.5	0.3	-
Q1.3	0.2	0.4	18.5	0.3	0.3	+280	0.6	0.2	-
Q2.1	0.3	1.1	18.1	0.4	2.7	+280	0.6	9.1	-
Q2.2	0.3	0.8	16.3	0.4	2.7	+280	0.7	8.2	-
Q2.3	0.3	0.8	15.2	0.4	2.2	+280	0.6	7.4	-
Q3.1	0.3	1.4	23.9	0.4	3.7	+280	0.8	15.1	-
Q3.2	0.6	0.7	18.5	0.8	0.7	+280	0.9	9.8	-

Q3.3	0.3	0.9	15.8	0.3	0.6	+280	0.7	3.7	-
Q3.4	0.2	0.6	15.9	0.2	0.2	+280	0.2	1.0	-
Q4.1	0.3	1.4	23.7	0.4	7.3	+280	0.7	14.7	-
Q4.2	0.3	1.0	23.3	0.4	2.0	+280	0.7	3.8	-
Q4.3	2.5	0.8	17.1	2.4	1.3	+280	2.9	2.0	-

Tabla 3. Resultados de las pruebas realizadas.

Según los resultados obtenidos, **Kylin es la herramienta que mejor rendimiento obtiene en tiempo o latencia de consulta**. Kylin logra una latencia de consulta inferior a 5 segundos en todas las pruebas y, en todas las consultas menos en la Q4.3, ese tiempo es inferior a 1 segundo.

En el caso de Vertica los resultados varían según la prueba. En las pruebas con 100M y 500M en la tabla de hechos, Vertica logra que casi todas las consultas se resuelvan en menos 5 segundos y, más de la mitad, en menos de 1 segundo.

Sin embargo, en la prueba más dura, con una tabla de hechos de 1.000 millones de filas, **Vertica logra unos resultados ligeramente peores que Kylin**, con dos consultas por encima de los 10 segundos. **Aun así, son tiempos muy buenos considerando el volumen de datos de la prueba** y que aún pueden aplicarse más optimizaciones en Vertica, como el uso de flattened tables o los ajustes de uso de memoria RAM por consulta (cuota disponible por usuario).

Como era de esperar, **PostgreSQL logra los peores resultados**. Aun así, en la primera prueba logra unos tiempos entre 15 y 25 segundos que no son especialmente malos teniendo en cuenta que se trata de una tecnología no Big Data. Sin embargo, en la prueba de 500M, las consultas son abortadas automáticamente por el sistema pasados unos 280 segundos sin respuesta.

Por ello, decidimos no llevar a cabo la prueba de 1.000M, ya que PostgreSQL no logró dar respuesta en un tiempo razonable en la prueba de 500M. Esto demuestra que Kylin y Vertica superan con bastante margen a un sistema tradicional como PostgreSQL, el cual sí hemos usado con éxito en muchas implantaciones de sistemas OLAP no Big Data.

En las siguientes imágenes se muestran los resultados por consulta en cada prueba realizada.

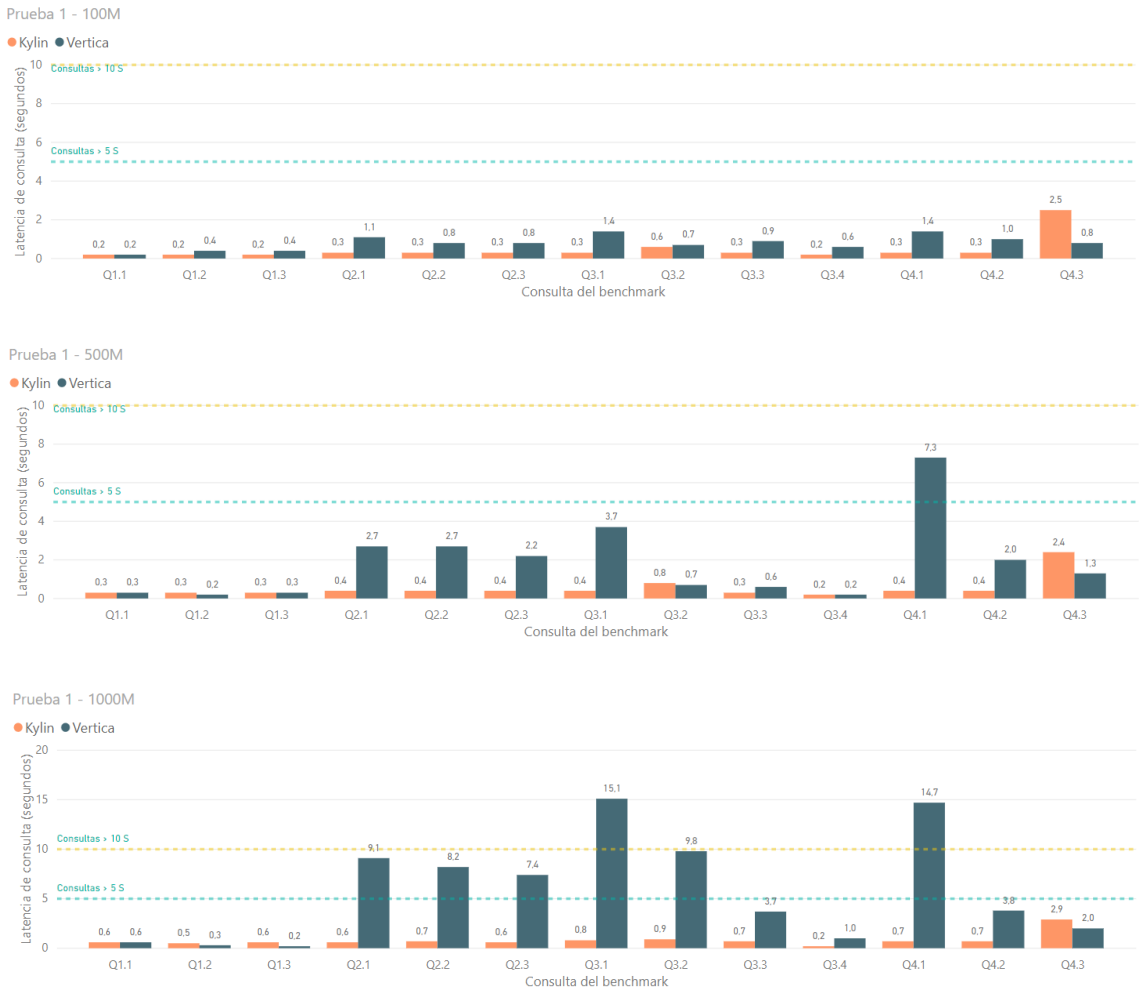


Figura 4. Latencias de cada consulta ejecutada para las 3 pruebas del benchmark.

En la Figura 5. puede observarse mejor la comparación de resultados entre Kylin y Vertica. En el caso de **Kylin, observamos que el tiempo o latencia de consulta apenas aumenta cuando se aumenta de forma considerable el tamaño del conjunto de datos**.

Esto se debe a que esta herramienta se basa en la pre-agregación de forma que, la computación y entrada/salida de disco necesarios en tiempo en tiempo consulta, son mucho menores que en el caso de Vertica. En el caso de Vertica observamos que la latencia media de consulta aumenta de forma considerable al aumentar el volumen de datos, especialmente en la última prueba de 1000M, la más agresiva. **No obstante, la media de 6 segundos de latencia de consulta que logra Vertica, sigue siendo un tiempo muy bueno y adecuado para un escenario Big Data OLAP.**

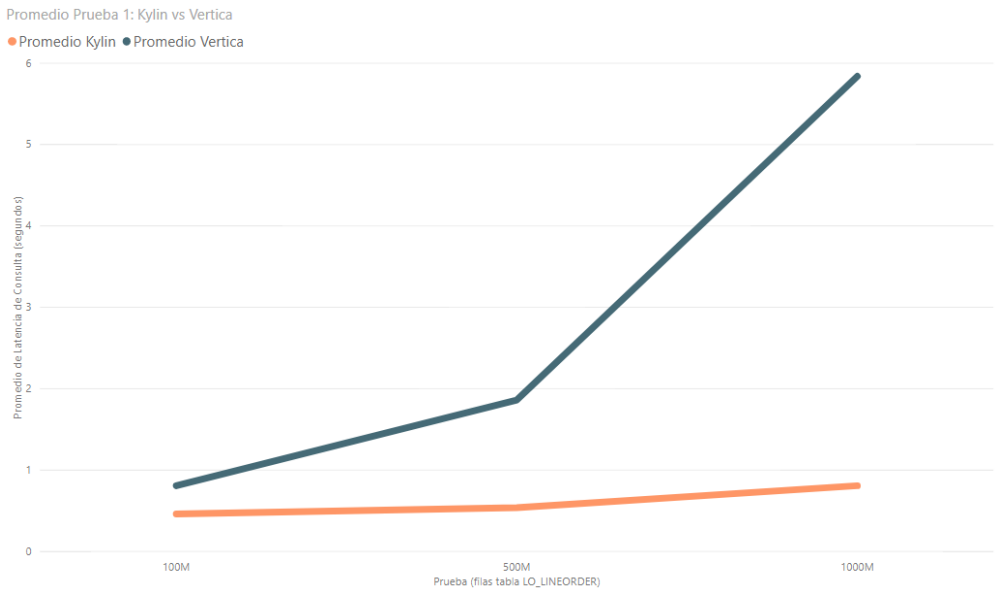


Figura 5. Relación entre tamaño de filas en la tabla de hechos y latencia de consulta entre Kylin y Vertica

También es necesario tener en cuenta otros factores, además de la latencia de consulta, para comparar Kylin y Vertica. **Un factor muy importante es el tiempo de carga de los datos.**

En el caso de Kylin la mayor parte de la computación se realiza en tiempo de carga (fase de construcción del cubo), para pre-agregar y pre-combinar los datos. Este es un proceso de tipo Hadoop - MapReduce que requiere disponer de importantes recursos hardware en el clúster Hadoop para reducir el tiempo de carga.

En Kylin la carga de todo el histórico de la prueba de 1.000M filas nos llevó unas 16 horas, un tiempo bastante elevando teniendo en cuenta que en Vertica cargamos todos los datos en cerca de 2 horas y con un consumo de recursos del clúster de Vertica menor (80% de recursos del clúster Hadoop, frente a un 40% en el clúster Vertica). No obstante, en Kylin es posible añadir o recargar datos de periodos como días o semanas en unos pocos minutos, un tiempo muy razonable para la carga incremental de datos.

Además, las últimas versiones de Kylin ya permiten la construcción o recarga de los datos usando Spark en lugar de Map Reduce, dando lugar a procesos de construcción más eficientes que pueden acelerar mucho el proceso de carga.

4. CONCLUSIONES

En este benchmark hemos puesto a prueba el rendimiento de consulta (latencia de consulta) de dos de las tecnologías Big Data OLAP más potentes del mercado, Kylin y Vertica. Además, hemos comparado estas tecnologías Big Data OLAP con una tecnología de base de datos relacional no Big Data, PostgreSQL.

El resultado del benchmark es que, **tanto Kylin como Vertica, son dos tecnologías adecuadas para escenarios Big Data OLAP** dónde queremos poder ejecutar consultas con respuesta interactiva sobre modelos analíticos con tablas de hasta varios miles de millones de filas.

Sin embargo, **los resultados de las pruebas han demostrado que Kylin es sensiblemente más rápido en consulta que Vertica, especialmente en la prueba de 1.000.000.000 filas (1000M) en la tabla de hechos LINEORDER.**

Además, Kylin a diferencia de Vertica, es una herramienta 100% open source sin ningún tipo de limitación, como sí ocurre con la versión gratuita de Vertica, limitada en número de máquinas del clúster (3) y almacenamiento (1TB). Si bien en muchos proyectos no es necesario superar estas limitaciones.

No obstante, Vertica ha obtenido resultados muy cercanos a Kylin en las 2 primeras pruebas y, para la tercera prueba, aún sería posible aplicar múltiples optimizaciones para lograr mejores resultados.

En cualquier caso, Vertica logra unos tiempos muy buenos que hacen que sea una alternativa a Kylin, más sencilla de instalar, con menores requerimientos hardware y que no requiere de un clúster Hadoop para funcionar.

Esto puede ser una ventaja en muchos proyectos dónde no se requieren otras herramientas del entorno Hadoop (ej. Spark o Kafka) para procesar cualquier tipo de fuente de datos Big Data, las cuales también requieren de mayores conocimientos y mantenimiento por parte de nuestro equipo de IT.

5. ANEXO 1: CONSULTAS BENCHMARK SSB

Q1.1

```
select sum(v_revenue) as revenue
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
where d_year = 1993
and lo_discount between 1 and 3
and lo_quantity < 25
```

Q1.2

```
select sum(v_revenue) as revenue
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
where d_yearmonthnum = 199401
and lo_discount between 4 and 6
and lo_quantity between 26 and 35;
```

Q1.3

```
select sum(v_revenue) as revenue
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
where d_weeknuminyear = 6 and d_year = 1994
and lo_discount between 5 and 7
and lo_quantity between 26 and 35;
```

Q2.1

```
select sum(lo_revenue) as lo_revenue, d_year, p_brand
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
```

```
left join SSB.part on lo_partkey = p_partkey
left join SSB.supplier on lo_suppkey = s_suppkey
where p_category = 'MFGR#12' and s_region = 'AMERICA'
group by d_year, p_brand
order by d_year, p_brand;
```

Q2.2

```
select sum(lo_revenue) as lo_revenue, d_year, p_brand
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
left join SSB.part on lo_partkey = p_partkey
left join SSB.supplier on lo_suppkey = s_suppkey
where p_brand between 'MFGR#2221' and 'MFGR#2228' and s_region = 'ASIA'
group by d_year, p_brand
order by d_year, p_brand;
```

Q2.3

```
select sum(lo_revenue) as lo_revenue, d_year, p_brand
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
left join SSB.part on lo_partkey = p_partkey
left join SSB.supplier on lo_suppkey = s_suppkey
where p_brand = 'MFGR#2239' and s_region = 'EUROPE'
group by d_year, p_brand
order by d_year, p_brand;
```

Q3.1

```
select c_nation, s_nation, d_year, sum(lo_revenue) as lo_revenue
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
left join SSB.customer on lo_custkey = c_custkey
left join SSB.supplier on lo_suppkey = s_suppkey
```

```
where c_region = 'ASIA' and s_region = 'ASIA'and d_year >= 1992 and d_year <= 1997
group by c_nation, s_nation, d_year
order by d_year asc, lo_revenue desc;
```

Q3.2

```
select c_city, s_city, d_year, sum(lo_revenue) as lo_revenue
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
left join SSB.customer on lo_custkey = c_custkey
left join SSB.supplier on lo_suppkey = s_suppkey
where c_nation = 'UNITED STATES' and s_nation = 'UNITED STATES'
and d_year >= 1992 and d_year <= 1997
group by c_city, s_city, d_year
order by d_year asc, lo_revenue desc;
```

Q3.3

```
select c_city, s_city, d_year, sum(lo_revenue) as lo_revenue
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
left join SSB.customer on lo_custkey = c_custkey
left join SSB.supplier on lo_suppkey = s_suppkey
where (c_city='UNITED KI1' or c_city='UNITED KI5')
and (s_city='UNITED KI1' or s_city='UNITED KI5')
and d_year >= 1992 and d_year <= 1997
group by c_city, s_city, d_year
order by d_year asc, lo_revenue desc;
```

Q3.4

```
select c_city, s_city, d_year, sum(lo_revenue) as lo_revenue
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
left join SSB.customer on lo_custkey = c_custkey
```

```
left join SSB.supplier on lo_suppkey = s_suppkey
where (c_city='UNITED KI1' or c_city='UNITED KI5') and (s_city='UNITED KI1' or s_city='UNITED KI5') and d_yearmonth = 'Dec1997'
group by c_city, s_city, d_year
order by d_year asc, lo_revenue desc;
```

Q4.1

```
select d_year, c_nation, sum(lo_revenue) - sum(lo_supplycost) as profit
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
left join SSB.customer on lo_custkey = c_custkey
left join SSB.supplier on lo_suppkey = s_suppkey
left join SSB.part on lo_partkey = p_partkey
where c_region = 'AMERICA' and s_region = 'AMERICA' and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
group by d_year, c_nation
order by d_year, c_nation;
```

Q4.2

```
select d_year, s_nation, p_category, sum(lo_revenue) - sum(lo_supplycost) as profit
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
left join SSB.customer on lo_custkey = c_custkey
left join SSB.supplier on lo_suppkey = s_suppkey
left join SSB.part on lo_partkey = p_partkey
where c_region = 'AMERICA'and s_region = 'AMERICA'
and (d_year = 1997 or d_year = 1998)
and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
group by d_year, s_nation, p_category
order by d_year, s_nation, p_category;
```

Q4.3

```
select d_year, s_city, p_brand, sum(lo_revenue) - sum(lo_supplycost) as profit
```

```
from SSB.p_lineorder
left join SSB.dates on lo_orderdate = d_datekey
left join SSB.customer on lo_custkey = c_custkey
left join SSB.supplier on lo_suppkey = s_suppkey
left join SSB.part on lo_partkey = p_partkey
where c_region = 'AMERICA'and s_nation = 'UNITED STATES'
and (d_year = 1997 or d_year = 1998)
and p_category = 'MFGR#14'
group by d_year, s_city, p_brand
order by d_year, s_city, p_brand;
```

6. SOBRE STRATEBI

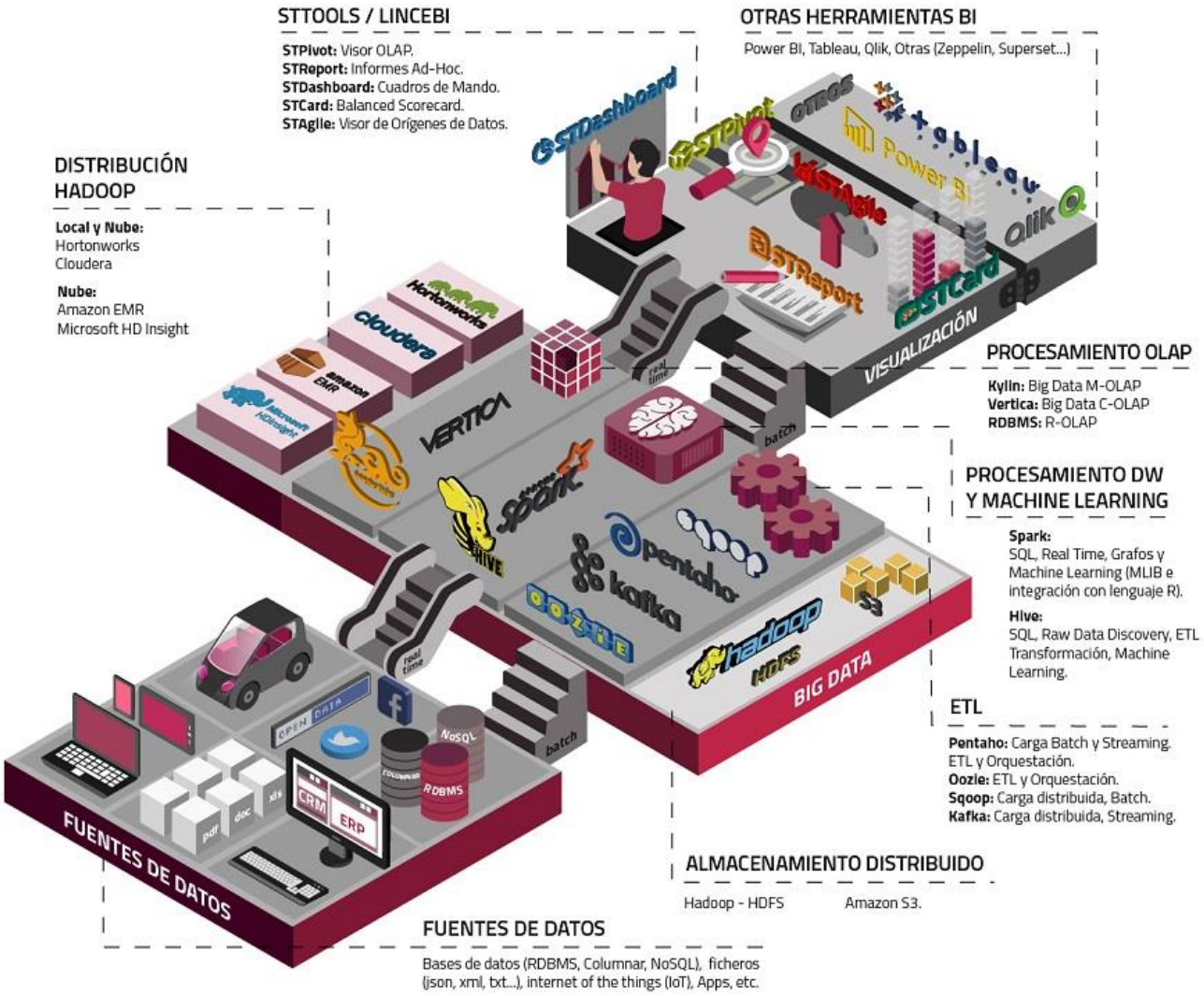
Stratebi es una empresa española, con oficinas en Madrid, Barcelona, Sevilla y Alicante, creada por un grupo de profesionales con amplia experiencia en sistemas de información, soluciones tecnológicas y procesos relacionados con soluciones de Open Source y de inteligencia de Negocio.

www.stratebi.com

Nuestra experiencia en Big Data se refleja en la implantación de **soluciones basadas en distribuciones Hadoop** con herramientas como Spark, Hive o Kafka, **sistemas OLAP como Kylin, Vertica o Amazon Redshift u otras NoSQL como Neo4J** y en sectores tan diversos como el marketing digital, retail, industria farmacéutica, turismo, educación, ciberseguridad o aeronáutica.

Además, complementamos estas soluciones **con herramientas ETL (Pentaho y Talend) y las herramientas de BI (Pentaho, Power BI o Tableau),** en las cuales tenemos una amplia experiencia en proyectos con o sin fuentes de datos Big Data. También impartimos formación en todas estas tecnologías y somos **partners oficiales de algunas de ellas como Hortonworks, Vertica o Talend**.

Stratebi ha creado la solución Big Data Analytics LinceBI.com, basada en tecnologías Open Source

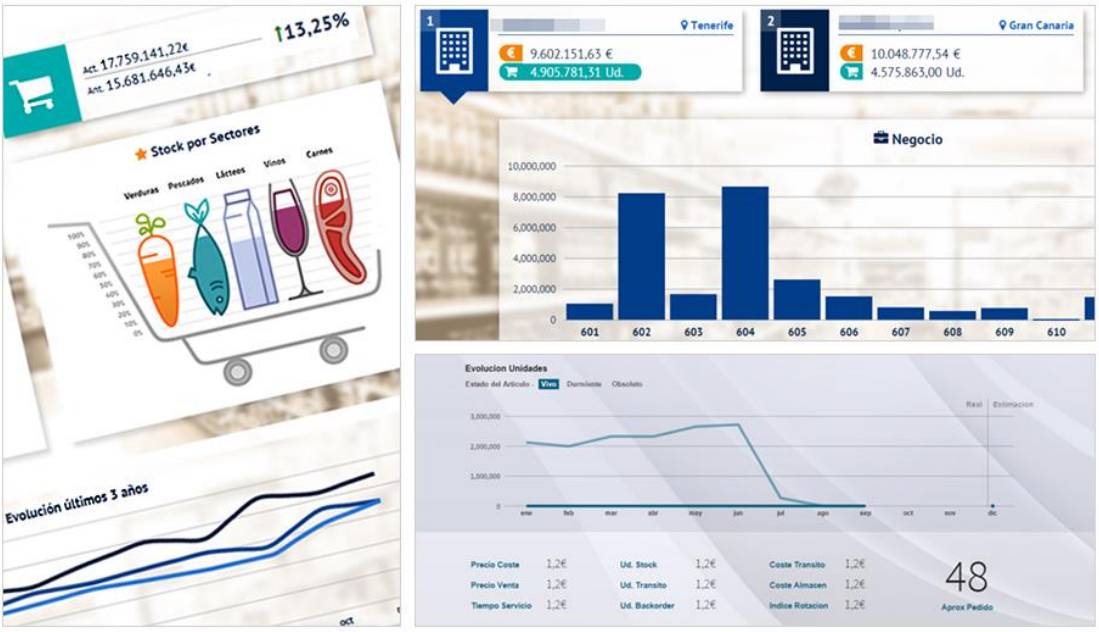

STTOOLS / LINCEBI
STPivot: Visor OLAP.
STReport: Informes Ad-Hoc.
STDashboard: Cuadros de Mando.
STCard: Balanced Scorecard.
STAgile: Visor de Orígenes de Datos.
OTRAS HERRAMIENTAS BI
Power BI, Tableau, Qlik, Otras (Zeppelin, Superset...)
DISTRIBUCIÓN HADOOP
Local y Nube:
Hortonworks
Cloudera
Nube:
Amazon EMR
Microsoft HD Insight
VISUALIZACIÓN
PROCESAMIENTO OLAP
Kylin: Big Data M-OLAP
Vertica: Big Data C-OLAP
RDBMS: R-OLAP
PROCESAMIENTO DW Y MACHINE LEARNING
Spark:
SQL, Real Time, Grafos y Machine Learning (MLIB e integración con lenguaje R).
Hive:
SQL, Raw Data Discovery, ETL Transformación, Machine Learning.
ETL
Pentaho: Carga Batch y Streaming. ETL y Orquestación.
Oozie: ETL y Orquestación.
Sqoop: Carga distribuida, Batch.
Kafka: Carga distribuida, Streaming.
BIG DATA
FUENTES DE DATOS
ALMACENAMIENTO DISTRIBUIDO
Hadoop - HDFS
Amazon S3.
FUENTES DE DATOS
Bases de datos (RDBMS, Columnar, NoSQL), ficheros (json, xml, txt...), internet of the things (IoT), Apps, etc.

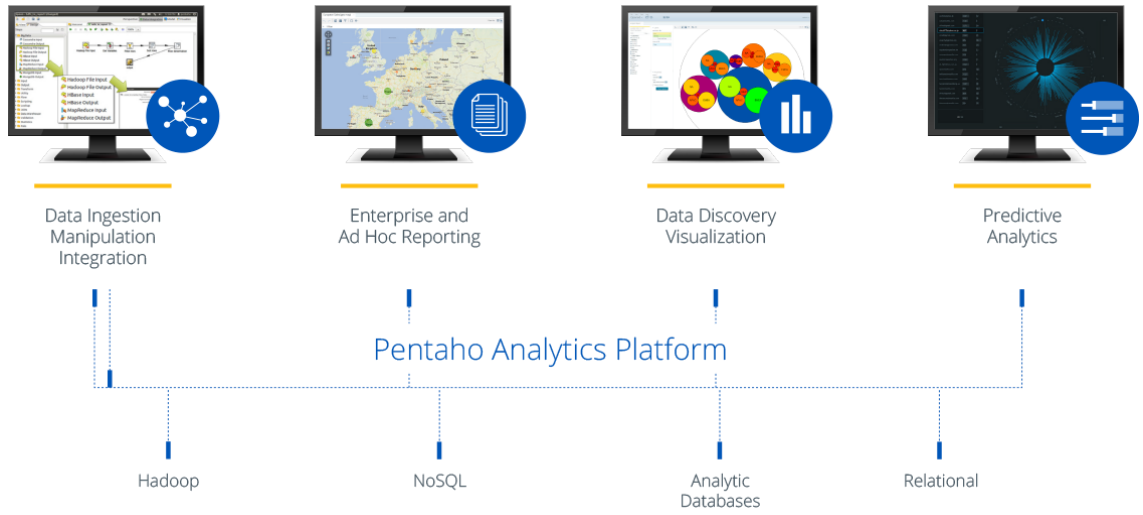
7. TECNOLOGÍAS

Recientemente, hemos sido nombrados Partners Certificados de Vertica, Talend y Microsoft

8. EJEMPLOS DE DESARROLLOS ANALYTICS

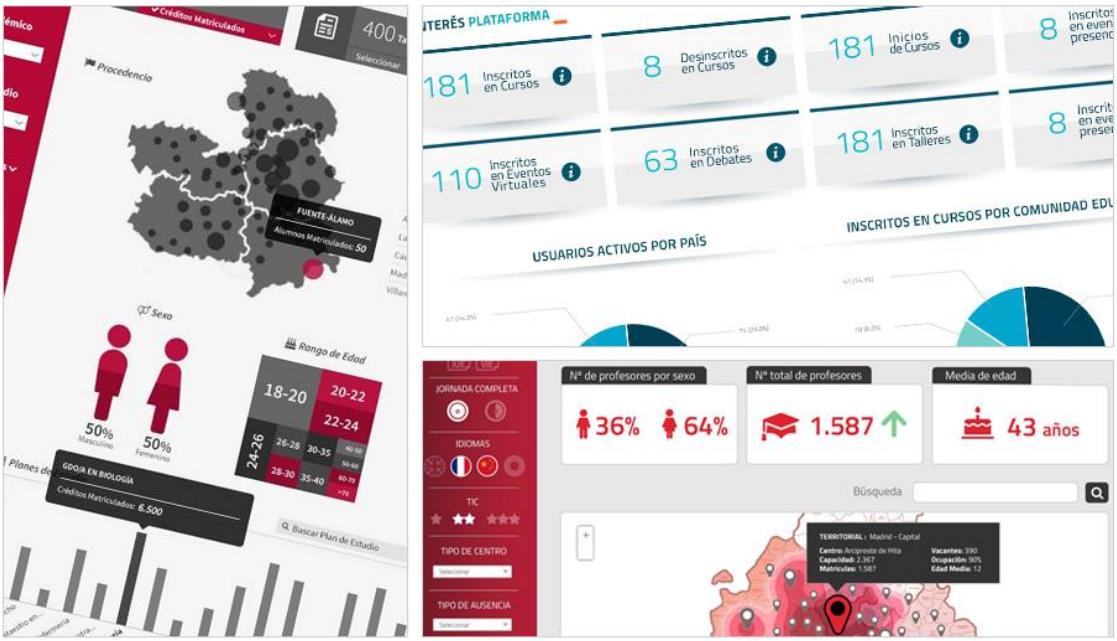
A continuación se presentan **ejemplos de algunos screenshots** de cuadros de mando y aplicaciones diseñados por Stratebi, con el fin de dar a conocer lo que se puede llegar a obtener, así como Demos Online en la web de Stratebi:

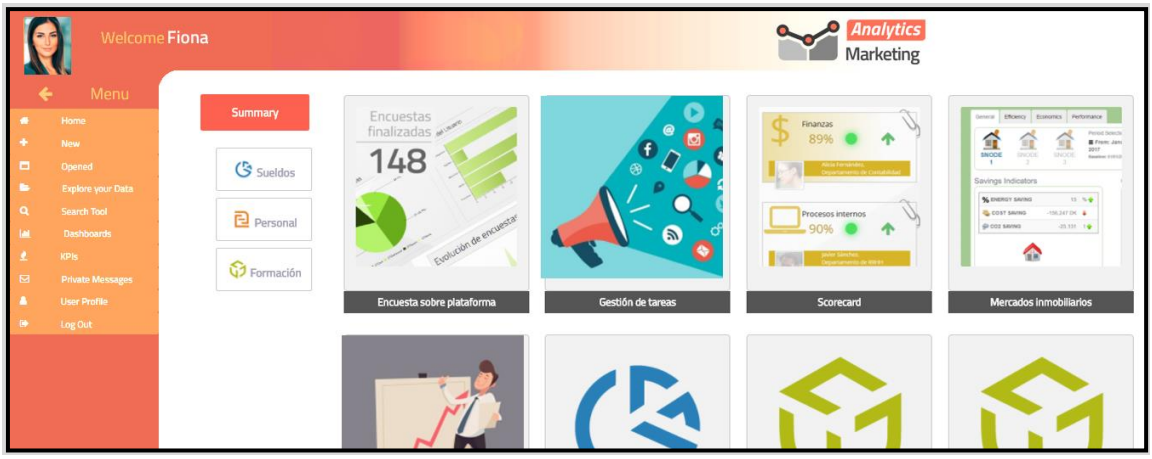





facebook DASHBOARD COMENTARIOS
Sexo
Sentimiento
Localización
65%
35%
28%
60%
12%
Tipo de recurso
Evolución Comentarios por días


smart MADRID
Último Actualización
18:24
TOP 5 PUNTOS DE MEDICIÓN
PROXIMIDAD DE PUNTOS DE MEDIDA
EVOLUCIÓN INTENSIDAD DE TRÁFICO


Procedencia
FUENTE-ÁLAMO
Sexo
50%
50%
Rango de Edad
18-20
20-22
22-24
24-26
26-28
28-30
30-35
35-40
GIDO/A EN BIOLOGÍA
Créditos Matriculados: 6.500
181 Inscritos en Cursos
8 Desinscritos en Cursos
181 Inicios de Cursos
110 Inscritos en Eventos Virtuales
63 Inscritos en Debates
181 Inscritos en Talleres
USUARIOS ACTIVOS POR PAÍS
INSCRITOS EN CURSOS POR COMUNIDAD EDU
JORNADA COMPLETA
IDIOMAS
TIC
TIPO DE CENTRO
TIPO DE AUSENCIA
Nº de profesores por sexo
36%
64%
Nº total de profesores
1.587
Media de edad
43 años
Búsqueda


Welcome Fiona
Analytics
Marketing
Menu
Home
New
Opened
Explore your Data
Search Tool
Dashboards
KPIs
Private Messages
User Profile
Log Out
Summary
Sueldos
Personal
Formación
Encuestas finalizadas
148
Evolución de encuestas
Encuesta sobre plataforma
Gestión de tareas
Finanzas
89%
Procesos internos
90%
Scorecard
Mercados inmobiliarios