Actualidad en *Cloud*

PID_00247338

Màrius Montón Macián

Tiempo mínimo de dedicación recomendado: 1 hora

Universitat Oberta de Catalunya

Índice

Introducción

En este contenido, se verán las principales tecnologías y soluciones que se pueden encontrar disponibles actualmente en *Cloud*.

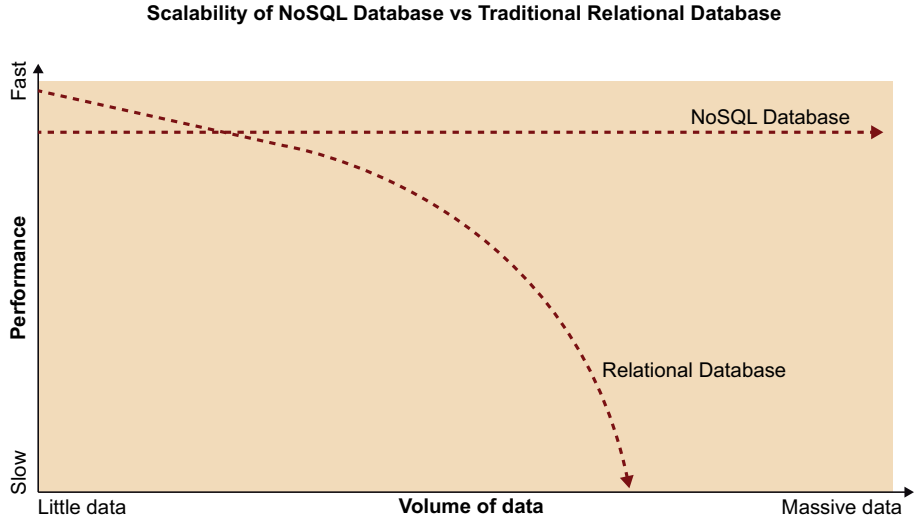
1. Servicios *Cloud*

En este apartado se presentan los servicios *Cloud* más habituales actualmente, que son las bases de datos para almacenar grandes cantidades de datos y los sistemas de ficheros masivos.

1.1. Bases de datos

Con el tiempo, la generación de datos se ha ido incrementando de forma exponencial. Esta generación tan elevada de datos ha producido la demanda de bases de datos que puedan almacenar un número ingente de datos de forma ágil y segura. Otras características que hasta ahora han sido primordiales en las bases de datos relacionales, como la integridad de los datos, penalizan de manera muy severa el rendimiento cuando se trata con una cantidad ingente de datos (figura 1). Además, debido a la gran cantidad de datos, estrategias clásicas de *backup* dejan de ser factibles por el gran coste que significarían.

Figura 1. Rendimiento de bases de datos tradicionales y NoSQL



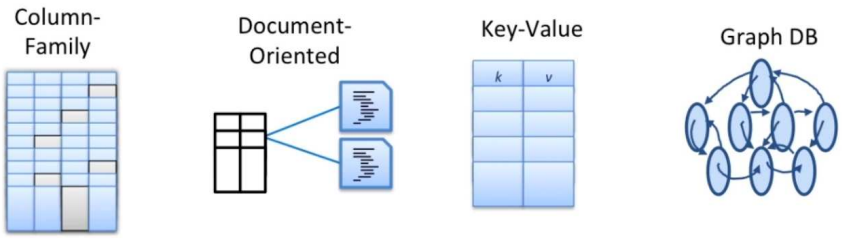
Por ello, aparecieron nuevos tipos y estrategias de bases de datos, con el objetivo de almacenar grandes volúmenes de datos de forma rápida, sencilla y segura.

Estas nuevas bases de datos normalmente siguen una estrategia NoSQL. Las bases de datos NoSQL tienen un modelo de almacenar y recuperar los datos distinto de las bases de datos relacionales clásicas y que se han usado a lo largo de los últimos 30 años (figura 2). Esta nueva aproximación permite el escalado horizontal de forma muy sencilla, lo que mejora el rendimiento general y la seguridad de los datos almacenados.

A cambio, puede verse comprometida la consistencia de los datos (es decir, que cada acceso a un dato reciba el último valor adscrito a ese dato), cosa que las bases de datos tradicionales suelen garantizar de forma total. Esta posible falta de consistencia es habitualmente temporal, ya que los cambios de un valor en un dato se propagan de forma rápida entre todos los componentes de la base de datos.

La mayoría de este tipo de bases de datos no guardan los datos en formato tabla como las bases de datos relacionales clásicas, sino que se usan almacenamientos del tipo <*clave, dato*> o basados en documentos (en los que cada dato es un documento completo). Otras bases de datos almacenan los datos en forma de documentos, y a lo que se permite el acceso es a estos documentos en conjunto (no a datos dentro de los documentos).

Figura 2. Tipos de bases de datos No-SQL



Elaboración propia

Estas bases de datos soportan migración horizontal, es decir, añadir una o varias máquinas nuevas incluso «en caliente», esto es, sin necesidad de apagar o suspender las operaciones de la base de datos. Así, si una base de datos está llegando al punto de saturación por la cantidad de datos o el número de consultas o accesos que se le hacen, es posible añadir más máquinas, de manera sencilla, para resolver el problema. De igual forma, es posible reducir el número de máquinas y redistribuir los datos en las máquinas restantes, para ajustarse a un descenso de las necesidades.

A continuación, se presentan las bases de datos de este tipo más habituales actualmente (primavera 2017).

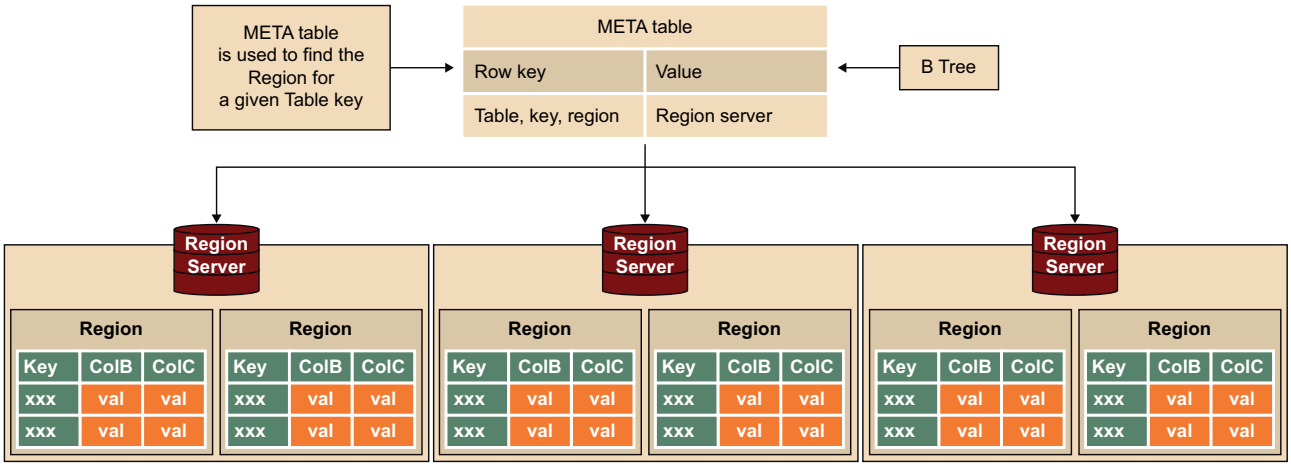
1.1.1. Cassandra

Proyecto de código libre de Apache Software Foundation, inicialmente creado dentro de Facebook [The apache foundation (2016)]. Base de datos descentralizada (ningún nodo es más importante que otro), soporta replicación (escalado horizontal). Puede tener problemas de inconsistencia de datos en caso de modificaciones simultáneas. Los datos se almacenen en un formato <*clave, dato*>, en el que un dato puede ser una tabla de datos completa.

1.1.2. HBase

Muy similar a lo anterior, también proyecto de código libre, orientada a grandes cantidades de datos y con el mismo estilo de tipo de datos [The apache foundation (2017)]. La diferencia principal es que HBase puede mantener la consistencia de datos, y las distintas máquinas en las que están distribuidos los datos tienen cierta jerarquía, es decir, algunas máquinas son más importantes y críticas que otras (véase figura 3).

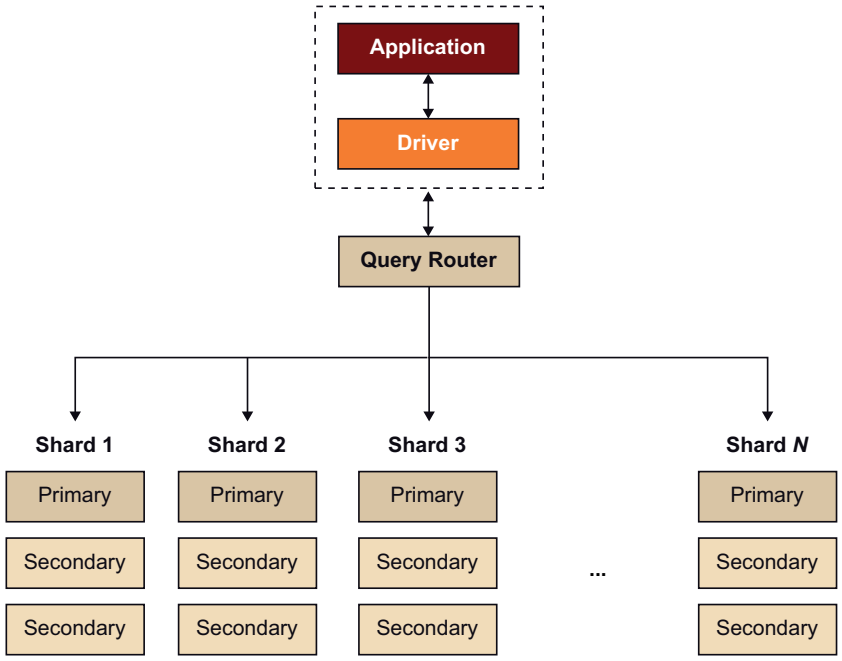
Figura 3. Arquitectura HBase



1.1.3. MongoDB

Es una base de datos también de código abierto, orientada al almacenamiento de datos tipo documento, esto es, una clave asociada a un fichero tipo JSON [MongoDB Inc. (2017)]. Al igual que la mayoría de las bases de datos NoSQL, MongoDB tampoco mantiene la integridad de los datos.

Figura 4. Arquitectura MongoDB



Como muchas más, permite replicación de los datos en distintos *shards* para mejorar el rendimiento y proteger los datos de fallos de hardware (figura 4).

1.1.4. Oracle NoSQL Database

Base de datos NoSQL orientada al almacenamiento masivo de datos tipo <*clave, dato*>, de la empresa Oracle [Oracle (2017)]. Ofrece las características típicas ya vistas en las otras bases de datos. Como diferencia, su soporte a la integridad de datos es completo.

1.1.5. BigTable

Base de datos de Google [Google Inc. (2017)] que almacena datos del tipo <*fila, columna, marca de tiempo*> y asocia esta clave con un conjunto de datos arbitrarios. Desde el principio, se diseñó para trabajar con enormes cantidades de datos (del orden de petabytes), y los datos se distribuyen en miles de servidores.

1.2. Sistemas de ficheros masivos

Hasta aquí, hemos presentado las bases de datos más usuales en el mundo *Cloud* e introducido brevemente sus principales virtudes.

Otro uso masivo de datos es el de los sistemas de ficheros masivos. Estos sistemas de ficheros están diseñados para manejar gran número de ficheros de datos, y son accesibles por gran cantidad de usuarios de manera simultánea; o para ejercer como *backup* de los mismos.

En este tipo de aplicación, lo que se busca es poder acceder a múltiples ficheros de gran tamaño (del orden de gigabytes o terabytes), por parte de múltiples usuarios simultáneamente, y que el rendimiento global del sistema no se vea penalizado. Estos requisitos son claramente distintos de los sistemas de ficheros clásicos o habituales, donde normalmente pocos usuarios (incluso solo uno) acceden a pocos ficheros, y habitualmente solo a uno o muy pocos de forma simultánea.

Para conseguir este propósito, las estrategias son distintas, pero la mayoría incluye la distribución de los datos por diferentes máquinas interconectadas entre sí. Además, el tamaño de bloque habitual (la unidad mínima de información) es muy grande, del orden de decenas de megabytes (al contrario que los sistemas de ficheros habituales, que son del orden de pocos kilobytes).

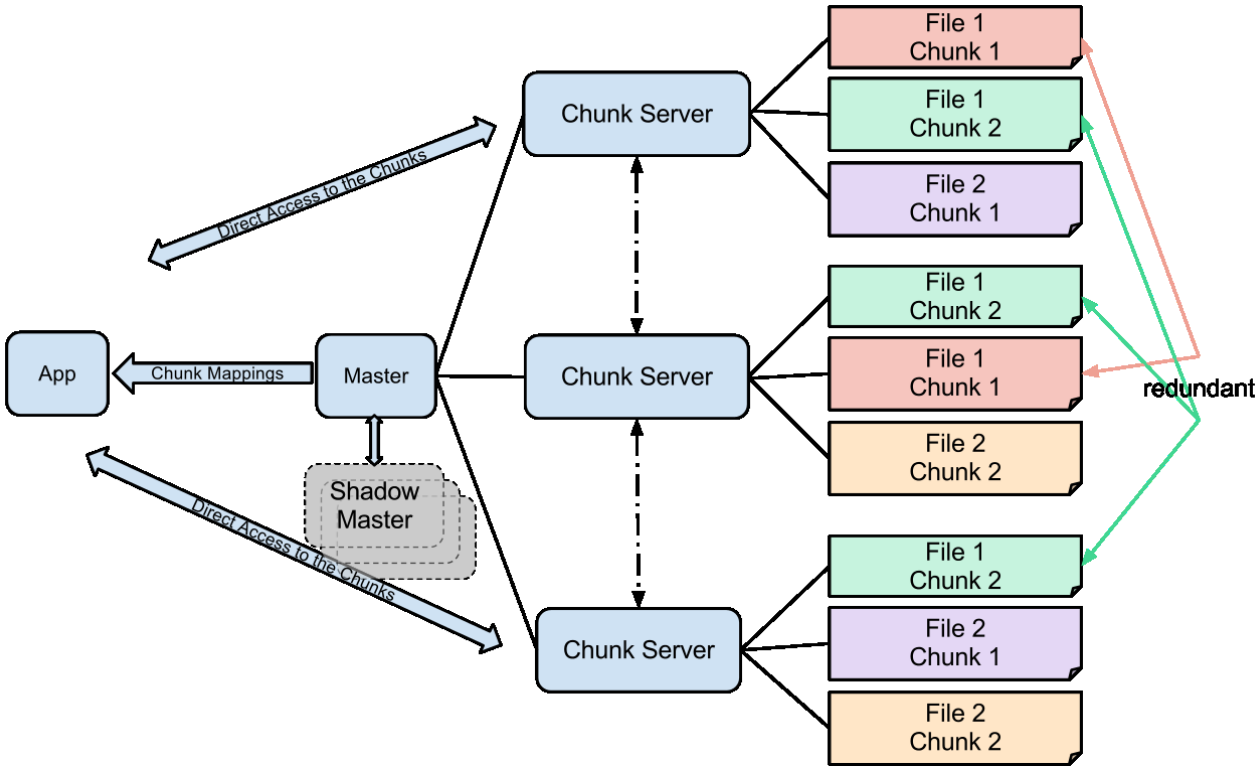
Por último, hay que destacar que habitualmente estos sistemas de ficheros están relacionados con otros servicios de tratamiento de datos tipo *big data*, como *MapReduce*.

Los sistemas de almacenamiento masivo más habituales se mencionan a continuación.

1.2.1. Google File System (GFS)

Sistema de ficheros creado por Google para sus servicios, que actualmente ofrece como componente de su IaaS. La información se centraliza en un servidor que reparte en distintas máquinas, según su disponibilidad, con una redundancia de al menos tres máquinas por cada dato. Así también aumenta su rendimiento, ya que distintos accesos al mismo dato pueden ser procesados por distintas máquinas (véase figura 5).

Figura 5. Diagrama de funcionamiento de GFS

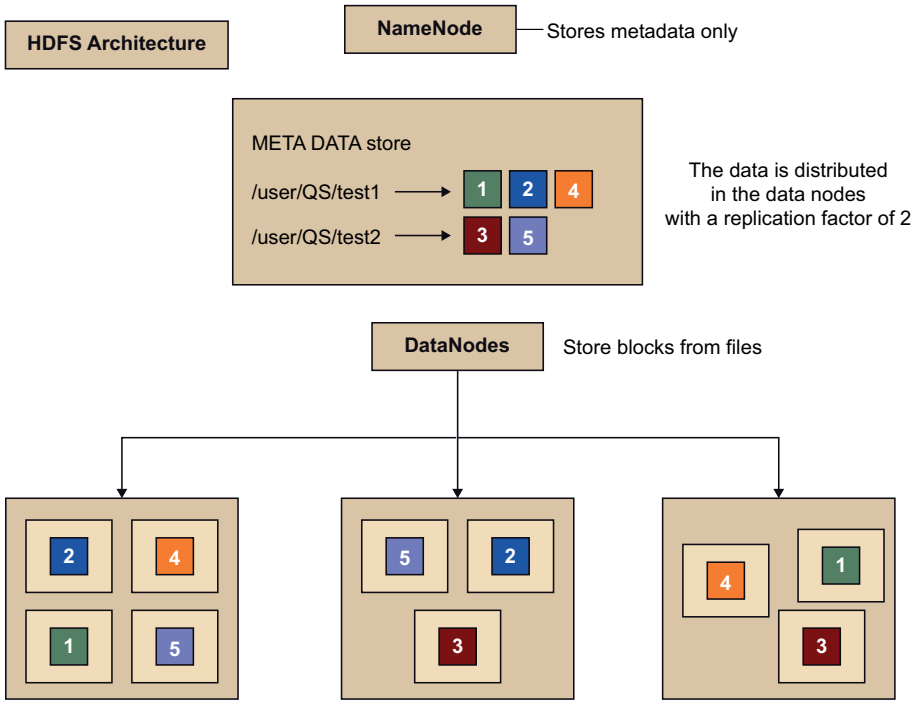


Por Saqibali [CC0], vía Wikimedia Commons

1.2.2. Hadoop Distributed File System (HDFS)

HDFS es uno de los componentes del sistema Hadoop. Como el anterior, los datos se replican en, al menos, tres máquinas distintas (véase figura 6).

Figura 6. Diagrama de funcionamiento de Hadoop Distributed File System



1.2.3. Amazon Glacier

Sistema de ficheros diseñado para datos a los que rara vez se vuelve a acceder, y que por tanto es ideal para tareas de *backup*. A cambio de un coste muy bajo por gigabyte al mes, los tiempos de recuperación de datos pueden llegar a ser de 5 horas (¡!).

2. *Cloud* de nueva generación

Con la proliferación de sistemas *Cloud*, su reducción de costes y su popularización en multitud de sistemas han facilitado un rápido desarrollo de nuevas ideas y tecnologías para este enfoque.

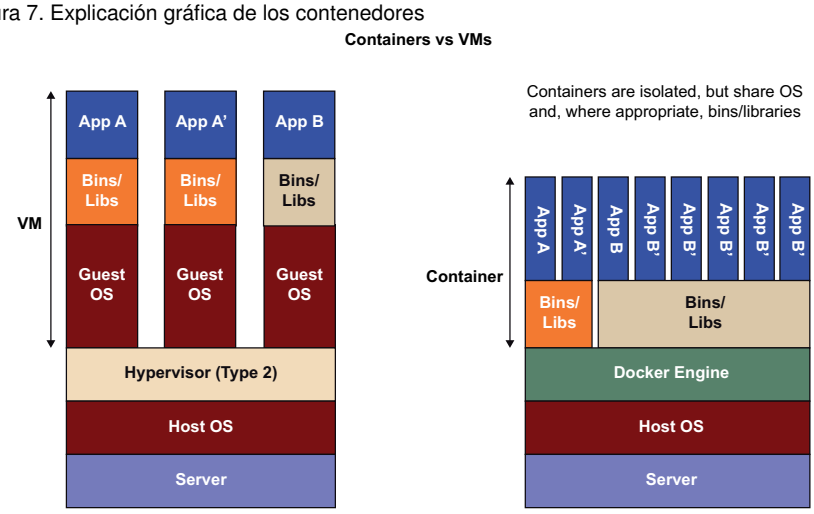
Así, aparte de poder utilizar de forma masiva grandes bases de datos y sistemas de ficheros para múltiples archivos y de gran tamaño (véase el apartado 1), se han desarrollado nuevas ideas y métodos de distinto tipo y con diferente enfoque.

2.1. Contenedores

Una faceta que ha tenido un rápido desarrollo en los últimos años ha sido la de las máquinas virtuales (véase el apartado 1). Por un lado, se han simplificado de forma notable las operaciones para arrancar y gestionar este tipo de software, y ahora es muy sencillo y rápido arrancar una VM en caso de que se necesite (por ejemplo, para el balanceo de carga). Por otro lado, el concepto de máquina virtual ha evolucionado hacia el de contenedores (*containers*, en inglés).

Un contenedor es un método de virtualización mas ligero que la máquina virtual tradicional, introducida en 1. Un contenedor es una imagen ligera con el software deseado y las librerías necesarias, sin incluir el sistema operativo ni el resto de las librerías y el software asociados. Con un contenedor así, un servicio de virtualización especial hace correr el software dentro del contenedor, de forma aislada del resto del sistema operativo, pero aprovechando las librerías nativas del ordenador huésped (véase figura 7).

Figura 7. Explicación gráfica de los contenedores



Las ventajas de esta técnica son, sobre todo, las siguientes:

- Portabilidad. Dado que se encapsula todo un software determinado junto con todas las librerías que necesita (sus dependencias), un contenedor podrá ejecutar correctamente el software que incluye en cualquier máquina que soporte contenedores.

- Rapidez. Puesto que se trata de una imagen ligera y que no hace falta cargar todo un sistema operativo entero cada vez que se carga la imagen, los tiempos de arrancar y parar un contenedor son muy rápidos (del orden de pocos segundos, contra decenas o centenares de segundos para una VM).

- Facilidad de integración. Ya que el contenedor se genera una sola vez y puede reutilizarse tantas veces como sea necesario, un equipo de desarrolladores puede trabajar con la misma versión de librerías y servicios y luego usar ese mismo contenedor para llevarlo a producción.

- Seguridad. Dado que el contenedor está limitado a las librerías instaladas dentro del mismo, y es supervisado por el sistema operativo huésped, los problemas de seguridad de un contenedor están limitados a lo que se pueda hacer dentro del contenedor, y el resto de los componentes quedan aislados.

- Ligero. Al instalar solo las librerías y el software necesarios, y no todo el sistema operativo, un contenedor ocupa mucho menos espacio que una máquina virtual entera.

El software de virtualización por contenedores más popular actualmente es **Docker**. Este software permite la creación, gestión y ejecución de contenedores de la forma explicada anteriormente. Desde su publicación como proyecto de código libre en el 2013, su crecimiento ha sido muy rápido y ya soportan de forma directa multitud de PaaS comerciales. De esta manera, es posible, en lugar de elegir un servicio en concreto, poner a ejecutar un contenedor de nuestra elección en el servicio PasS.

Algunos proyectos de software se encargan de gestionar contenedores de forma independiente del virtualizador software que los ejecutará después. Esto permite una mayor flexibilidad y facilidad de gestión, así como construir de manera sencilla *clusters* de contenedores. Los más populares son Swarm (de la misma compañía que desarrolla Docker) y Kubernetes.

Con estos gestores, es posible, por ejemplo, configurar el sistema para que, si la carga de un contenedor determinado sube demasiado (por tener demasiados clientes conectados simultáneamente), se ejecute otra imagen del mismo contenedor y un nuevo contenedor que se dedique a balancear a los usuarios entre las dos imágenes. Una vez configurado el gestor, el proceso se lleva a cabo de manera automática, sin necesidad de supervisión.

2.2. Microservicios

Otro cambio tecnológico que se ha visto propiciado por la popularización de los sistemas *Cloud* ha sido del de los microservicios (*microservices*, en inglés).

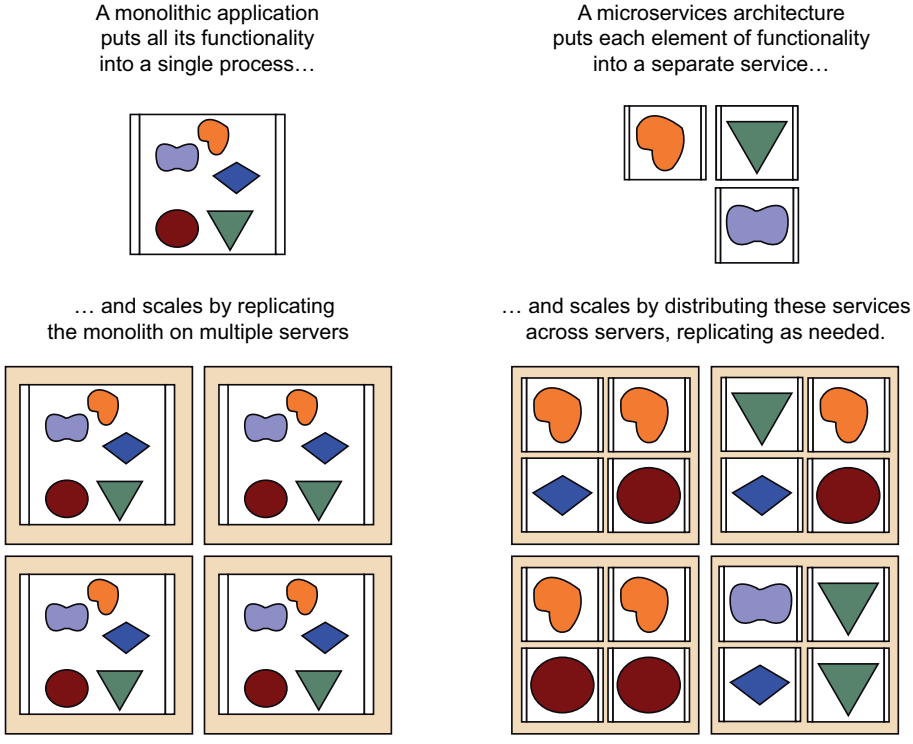
El cambio que supone la programación basada en microservicios es el siguiente:

> En lugar de tener una sola aplicación encargada de todas las tareas para solucionar un problema dado, este problema se trocea en subproblemas más sencillos y se diseñan pequeñas aplicaciones para solucionar cada una de las pequeñas partes en las que ha quedado dividido. Cada una de estas partes recibe el nombre de microservicio o servicio (véase figura 8).

Con esta estrategia se gana en modularidad, en facilidad de testear, claridad para los desarrolladores, escalabilidad y paralelización de desarrollo. Además, debido a que cada servicio es independiente de los demás, pueden desarrollarse en el lenguaje de programación que mejor se adecúe a ese problema en concreto.

Con la tecnología adecuada, estos servicios pueden ser intercambiados por otros mejorados, replicados varias veces para ganar en rendimiento, etc. Para esto, hace falta que todo el entorno esté bien diseñado, que los servicios se conecten entre ellos de una forma estándar e intercambien datos de manera transparente.

Figura 8. Explicación gráfica de los microservicios



Como se puede apreciar, estas características encajan perfectamente en el mundo *Cloud*, y es posible repartir estos servicios en distintas máquinas (ya sean reales, virtuales o contenedores) para optimizar en cualquier momento el rendimiento de todo el sistema y soportar fallos de distinto tipo (fallos de máquinas, fallos de código, ataques maliciosos, etc.).

3. Resumiendo

Por todo lo visto hasta ahora, el *Cloud* ofrece ventajas a las empresas respecto a la estrategia tradicional de mantener equipos propios.

Estas ventajas se pueden resumir en:

- Inversión de capital fijo en gasto variable: en lugar de invertir grandes sumas de dinero en centros de datos y servidores, se paga según los recursos que se consumen y por los recursos que se usen.

- Economía de escala: mediante el uso de *Cloud*, se puede lograr un coste variable más bajo de lo que puede obtener una empresa sola. Debido a que los proveedores *Cloud* agregan a cientos de miles de clientes, estos pueden lograr mayores economías de escala, lo que se traduce en bajos precios.

- Reducir incertezas: no son necesarias buenas previsiones de crecimiento de la infraestructura, ya que el escalado puede hacerse de manera automática y muy rápida.

- Mejores equipos: en un entorno *Cloud*, es muy fácil añadir nuevos recursos, lo que significa que reduce de semanas a minutos el tiempo que se tarda en poner estos recursos a disposición. Esto se traduce en un aumento dramático en la agilidad de la organización, ya que el coste y el tiempo necesarios para experimentar y desarrollarse son significativamente menores.

- Flexibilidad: debido a la migración horizontal y su automatización, es posible adaptarse a las necesidades reales en cada momento, añadiendo y quitando capacidad de proceso, cómputo o disco, según sea necesario.

- Concentración: la empresa puede centrarse en su negocio y no en mantener ni gestionar su infraestructura informática.

Bibliografía

Google Inc. (2017). «CLOUD BIGTABLE».
URL: «https://cloud.google.com/bigtable/».

MongoDB Inc. (2017). «MongoDB».
URL: «https://www.mongodb.com/».

Oracle (2017). «Oracle NoSQL Database».
URL: «https://www.oracle.com/database/nosql/index.html».

The apache foundation (2016). «Apache Cassandra».
URL: «http://cassandra.apache.org/».

The apache foundation (2017). «HBase».
URL: «https://hbase.apache.org/».