Introducción a las infraestructuras para *big data*

PID_00286215

Remo Suppi Boldrito

Tiempo mínimo de dedicación recomendado: 4 horas

Universitat Oberta de Catalunya

Remo Suppi Boldrito

Ingeniero de Telecomunicaciones. Doctor en Informática por la Universitat Autònoma de Barcelona (UAB). Profesor del Departamento de Arquitectura de Computadores y Sistemas Operativos en la UAB.

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Josep Jorba Esteve

Primera edición: febrero 2022
© de esta edición, Fundació Universitat Oberta de Catalunya (FUOC)
Av. Tibidabo, 39-43, 08035 Barcelona
Autoría: Remo Suppi Boldrito
Producción: FUOC

Los textos e imágenes publicados en esta obra están sujetos –excepto que se indique lo contrario– a una licencia Creative Commons de tipo Reconocimiento-Compartir igual (BY-SA) v.3.0. Se puede modificar la obra, reproducirla, distribuirla o comunicarla públicamente siempre que se cite el autor y la fuente (Fundació per a la Universitat Oberta de Catalunya), y siempre que la obra derivada quede sujeta a la misma licencia que la obra original. La licencia completa se puede consultar en: http://creativecommons.org/licenses/by-sa/3.0/es/legalcode.es

Índice

Objetivos

Los objetivos principales de este módulo son los siguientes:

1. Conocer los conceptos y arquitecturas básicas que permiten incrementar las prestaciones de un sistema de cómputo.

2. Conocer las principales estructuras y agrupaciones de una infraestructura de procesamiento de la información, definir cómo deben moverse los datos y saber con qué ventajas y desventajas cuenta cada una de ellas.

3. Caracterizar el concepto de *big data*, cuándo se aplica, qué características tiene y cómo se utiliza para extraer información basada en datos.

4. Analizar los modelos de procesamiento e interacción que se desarrollan para que los programadores puedan acceder a servicios distribuidos de forma estandarizada y segura (API).

Conceptos más importantes

El estudiantado deberá centrar su atención en los siguientes conceptos fundamentales presentados en este módulo:

- Prestaciones de una aplicación
- GPU
- Clúster
- Cómputo de altas prestaciones (*high performance computing*)
- Nube (*cloud*)
- *Big data*
- IoT (*Internet of things*)
- Cómputo en la periferia (*Edge computing*)
- API (*application programming interface*)

1. Incremento de las prestaciones

Existen diversas técnicas para aumentar las prestaciones de un programa informático, relacionadas con la arquitectura de procesador, como son, entre otras:

- la planificación dinámica de instrucciones,
- la predicción dinámica de los saltos y
- el *multithreading*.

Estas mejoras, acompañadas por el incremento del número de núcleos (*cores*), la programación en memoria compartida y los incrementos de memoria caché, permiten obtener unas prestaciones inmejorables desde el punto de vista de las aplicaciones. Pero ¿cómo debemos proceder cuando es necesario incrementar aún más estas prestaciones?

La idea de incrementar el **número de núcleos o *threads*** tiene un límite que depende del procesador, pero se podrán obtener mejores prestaciones si se incrementa el número de unidades de procesamiento (CPU) y se transforma el sistema en uno multinúcleo (*multicore*) con múltiples CPU.

El incremento del número de *threads* –incrementando el número de *cores*, si es posible, o utilizando las características de *hyper-threading* de los procesadores, si está disponible– está limitado por las características de la CPU, por lo que para determinados problemas es interesante contemplar opciones como las que ofrece una GPU.

> Una unidad de procesamiento gráfico (GPU, *graphics processing unit*) es un coprocesador dedicado al procesamiento de gráficos, pero que puede ser utilizado para ejecutar determinado tipo de aplicaciones de propósito general (GPGPU, *general-purpose computing on graphics processing units*).

Aunque una GPU es un procesador diseñado para el procesamiento de gráficos 3D interactivos, muchas de sus características (por ejemplo, bajo precio en relación con su potencia de cálculo, gran paralelismo, optimización para cálculos en coma flotante) resultan muy interesantes para incrementar las prestaciones de determinadas aplicaciones en el ámbito científico.

Existen diferencias sustanciales entre las **arquitecturas de una GPU y una CPU**, y debemos ser conscientes de que no todas las aplicaciones pueden beneficiarse de las ventajas que presenta una GPU. En concreto, el **acceso a memoria** es una de las principales dificultades, ya que las CPU están diseñadas

para el acceso aleatorio a memoria, lo cual favorece la creación de estructuras de datos complejas y su gestión mediante punteros. En cambio, en una GPU, el acceso a memoria está mucho más restringido.

Memoria en un procesador de vértices y uno de píxeles

En un procesador de vértices (la parte de una GPU diseñada para transformar vértices en aplicaciones 3D), se favorece el modelo de distribución, en el que el programa lee en una posición predeterminada de la memoria, pero escribe en una o varias posiciones arbitrarias. En cambio, un procesador de píxeles, o fragmentos, favorece el modelo de recolección, por lo que el programa puede leer en varias posiciones arbitrarias, pero escribir en solo una posición predeterminada.

Por ese motivo, el desarrollador de aplicaciones GPGPU debe adaptar los accesos a memoria y las estructuras de datos a las características de la GPU, utilizando, por ejemplo, el almacenamiento de datos en una memoria 2D (donde normalmente se almacenaría una textura); el acceso a esas estructuras de datos sería equivalente a una lectura o escritura de una posición en la textura, lo cual puede representar problemas porque no se puede leer y escribir en la misma textura, y si esa operación es imprescindible para el desarrollo del algoritmo, este se debe realizar en diferentes secuencias.

Debe tenerse en cuenta que aunque la mayoría de **algoritmos** que se pueden desarrollar en una CPU también pueden ser desarrollados en una GPU, ambas implementaciones no serán igual de eficientes en las dos arquitecturas. Los algoritmos con un alto grado de paralelismo, que no requieren estructuras de datos complejas y tienen una alta intensidad aritmética, son los que mayores beneficios obtienen de su implementación en la GPU y sus prestaciones son significativamente mejores que sus equivalentes en CPU.

En cuanto al **desarrollo del software GPGPU**, inicialmente se realizaba en lenguaje ensamblador o bien en alguno de los lenguajes específicos para aplicaciones gráficas (GLSL, Cg o HLSL), pero hoy en día se utiliza CUDA (Nvidia), una extensión de C que permite la codificación de algoritmos en GPU de Nvidia. También, y cada vez son más los programadores que lo hacen, se puede utilizar OpenCL, una combinación de interfaz y lenguaje de programación para el desarrollo de aplicaciones paralelas que puedan ser ejecutadas, de forma transparente, en diversas unidades de procesamiento (CPU multinúcleo, GPU, etc.).

2. Infraestructura HPC y computación en la nube

Para incrementar aún más las prestaciones, existen otras alternativas para el procesamiento de aplicaciones, que consisten en diferentes **organizaciones del sistema de información**. Entre las principales, se pueden mencionar las siguientes:

1) Clústeres: se aplica a agrupaciones de ordenadores unidos entre sí (generalmente) por una red de alta velocidad que se comportan como si fuesen un único ordenador. Son utilizados como **entornos de procesamiento de altas prestaciones** (HPC, *high performance computing*) y servicios para aplicaciones críticas o de alto rendimiento, entre otros usos. Su utilización se ha popularizado para aplicaciones que necesitan un elevado tiempo de cómputo (por ejemplo, las científicas) y están basados en infraestructura comercial (*blades* o *slices*) junto con una red de altas prestaciones (por ejemplo, Infiniband), lo cual, gracias al uso en su gran mayoría de software de código abierto (Linux o NFS, aunque cada vez más está siendo reemplazado por otros sistemas de archivos distribuidos, como Ceph, Lustre, GlusterFS), de sistemas de gestión de colas (como SGE y Slurm) y de librerías (por ejemplo, OpenMPI para ejecutar tareas distribuidas sobre la arquitectura u OpenMP para aprovechar la potencialidad de los sistemas multicore), permite desarrollar y ejecutar aplicaciones concurrentes o distribuidas de altas prestaciones. Así se puede disponer de una infraestructura que provee un alto rendimiento, alta disponibilidad, con balanceo de carga, eficiente, escalable y a costes razonables.

Enlaces complementarios

Infiniband: <https://www.infinibandta.org>

SGE: <https://sourceforge.net/projects/ gridengine/>

Slurm: <http://slurm.schedmd.com>

OpenMPI: <https://www.open-mpi.org>

OpenMP: <https://www.openmp.org>

2) Superordenador: es una evolución funcional a gran escala de un clúster (aunque con diferentes arquitecturas), pero con una capacidad de cómputo mucho más elevada que este (y, por lo tanto, que un ordenador de propósito general). Su rendimiento se mide en teraflops (10^{12} operaciones de punto flotante por segundo).

Los superordenadores más potentes

En junio de 2020, el superordenador más potente, publicado en la lista del Top500 (los 500 superordenadores más potentes del mundo), era el Supercomputer Fugaku, que se encuentra en el RIKEN Center for Computational Science, en Japón; y se mantiene como número uno en la última lista publicada en noviembre de 2021. En el 2020 tenía 7.299.072 núcleos, una potencia de 415.530 teraflops y consumía 28.335 kW, y en el 2021 ha incrementado el número de *cores* a 7.630.848, con una potencia de 442.010 teraflops y un consumo de 29.899 kW. En Cataluña se encuentra el Marenostrum (la versión actual, el Marenostrum 4, que estaba en el 2020 en la posición 63 de la lista y en noviembre de 2021 ha pasado a la posición 73, pero ya está en fase de actualización al Marenostum 5).

Enlaces complementarios

Lista del Top500: <https://www.top500.org/lists/top500>

Barcelona Supercomputing Center (Marenostrum): <https://www.bsc.es>

Actualización al Marenostum 5: <https://www.bsc.es/news/bsc-news/marenostrum-5-will-host-experimental-platform-create-supercomputing-technologies-%E2%80%9Cmade-europe%E2%80%9D>

Es importante mencionar que un superordenador puede estar basado en dos enfoques diferentes:

Enlaces complementarios

Centro de Supercomputación de Galicia (Finisterrae): <https://www.cesga.es>

Juqueen: <https://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/de-installedSystems/JUQUEEN/Configuration/Configuration.html>

a) La **evolución del clúster** (como, por ejemplo, el Marenostrum en Cataluña, el Finisterrae en Galicia o el Juqueen en Alemania), en la que los clústeres se utilizan de forma dedicada para el cómputo de altas prestaciones, comparten un espacio físico y disponen de una serie de recursos comunes, como redes de comunicación de alta velocidad o un espacio de almacenamiento compartido y distribuido.

b) La **distribución a través de una red** formada por miles o cientos de miles de ordenadores discretos conectados mediante Internet, que se dedican de forma no exclusiva a la solución de un problema común (generalmente, cada equipo recibe y procesa un conjunto de tareas pequeñas y traslada los resultados a un servidor central, que los integra en una solución global).

Distribución a través de una red

Son representativos de distribución a través de una red proyectos como Seti@home (http://setiathome.ssl.berkeley.edu), Folding@home (https://foldingathome.org), World Community Grid (http://www.worldcommunitygrid.org) o ClimatePrediction.net (http://www.climateprediction.net), muchos de ellos basados en Boinc (una arquitectura software de código abierto que permite que voluntarios aporten sus recursos a un problema común –paradigma conocido como *volunteer computing*–); según la página de sus desarrolladores, la potencia de cómputo generada por esta arquitectura en todos los proyectos que la utilizan es de promedio en 24 horas: 30.286 petaflops; voluntarios activos: 86.116, con 933.264 computadoras (noviembre de 2020).

Enlaces complementarios

Boinc: <https://boinc.berkeley.edu>

Potencia de cómputo de Boinc: <https://boinc.berkeley.edu/computing.php>

3) ***Grid***: se refiere a una infraestructura que permite la integración y el uso colectivo de ordenadores de alto rendimiento, redes y bases de datos propiedad de diferentes instituciones, pero unidos por una capa de software (*middleware*) común que permite utilizarlos como si fuera un único superordenador. Con el fin de colaborar aportando los recursos de cómputo gestionados por cada institución, las universidades o laboratorios de investigación se asocian para formar *grids* y así ceder sus recursos temporalmente, pero también disponer del cómputo equivalente a la unión de todas estas infraestructuras cuando lo requieran.

La idea del *grid* fue establecida por Ian Foster y Carl Kesselman, los precursores en la creación de Globus Toolkit (http://toolkit.globus.org/toolkit), considerado como la primera herramienta para construir *grids* (hoy en día obsoleto y retirado).

Lectura complementaria

Ian Foster; Carl Kesselman (2014). *The History of the Grid* [en línea]. <http://www.ianfoster.org/wordpress/wp-content/uploads/2014/01/History-of-the-Grid-numbered.pdf>

Proyectos *grid*

Entre los grandes proyectos *grid* cabe citar CrossGrid, EU-DataGrid o, más recientemente, el proyecto EGI-InSPIRE (https://www.egi.eu/about/a-short-history-of-egi), todos ellos financiados por la Unión Europea.

Es importante destacar que el *grid* es un tipo de infraestructura de cómputo cuyo ámbito de utilización y propósito natural es el **científico (*e-science*)**, usado en universidades y laboratorios de investigación, mientras que la nube nace de la idea de la prestación de servicios para los negocios de empresas y usuarios

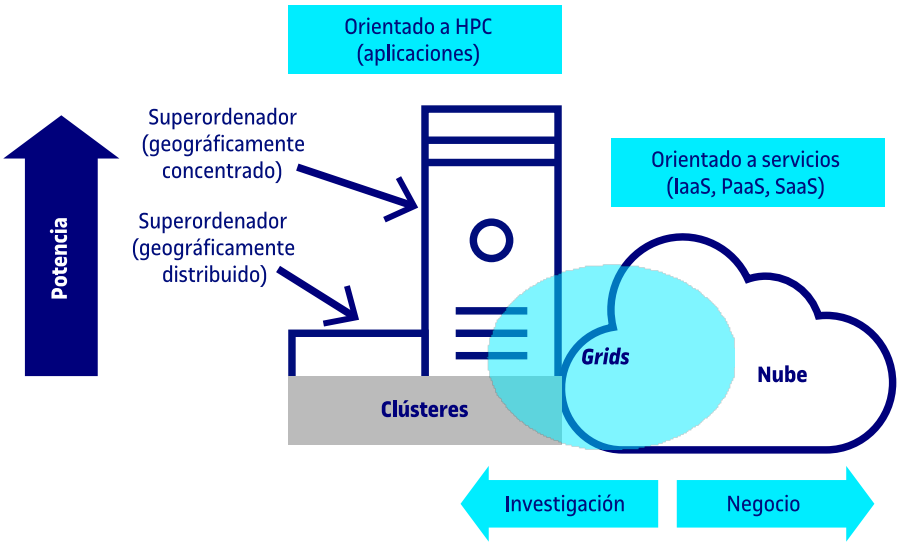
(*e-business*) desde otras empresas, pero ambos son similares en paradigmas y estructuras para manejar grandes cantidades de recursos distribuidos. Muchos expertos consideran estas dos vías como las mejores para el uso masivo de recursos (bajo diferentes modelos): una para la e-ciencia exclusivamente (*grid*) y otra para las empresas (nube). Es importante destacar que se comienzan a ofrecer **servicios cruzados**, es decir, altas prestaciones (HPC) en la nube.

HPC en la nube

AWS HPC (https://aws.amazon.com/hpc) es un ejemplo de altas prestaciones en la nube: instancias con base en hardware específico y de altas prestaciones.

Tras esta caracterización de los recursos y su utilización, podemos ubicar gráficamente cada una de estas arquitecturas en función de su potencia de cómputo y su tipo de dedicación (ver figura 1).

Figura 1. Arquitecturas en función de su potencia de cómputo y su tipo de dedicación



Como se verá en los siguientes módulos, la principal tecnología que da soporte a la computación en la nube (*cloud computing*) es la virtualización. La **capa de virtualización**, conocida también como *hipervisor*, separa un dispositivo físico en uno o más dispositivos «virtuales» (llamadas *máquinas virtuales*), que pueden ser asignados, utilizados y gestionados por el cliente de forma muy simple, como si de máquinas físicas se tratara, pero con las consiguientes ventajas (aislamiento, fácil mantenimiento, puesta en marcha, etc.). Hoy en día todos los sistemas utilizan técnicas de virtualización asistidas por hardware (extensiones del procesador VT-x o AMD-V), de forma que es posible tener máquinas virtualizadas muy eficientes y configuradas (mediante técnicas de clonado o *pool of VM in stand-by*) para ser asignadas a un usuario bajo demanda y en pocos segundos.

Hipervisores

Algunos ejemplos de hipervisores son KVM (Linux) (https://www.linux-kvm.org/page/Main_Page), VirtualBox (Linux, OS X, Windows y código abierto) (https://www.virtualbox.org), VMware ESXi (con un versión gratuita para uso no comercial) (https://www.vmware.com/products/esxi-and-esx.html) o Hyper-V (disponi-

ble para Windows 10) (https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/enable-hyper-v).

Un paso adicional que ha permitido un incremento notable de la eficiencia y la disponibilidad, lo que ha supuesto una revolución en los entornos de desarrollo, ha sido la virtualización a nivel del sistema operativo para la creación de un sistema escalable de múltiples entornos independientes con un mínimo impacto en la utilización de los recursos.

La virtualización del sistema operativo (SO) permite que su núcleo (*kernel*) pueda albergar múltiples instancias de usuario aisladas, donde cada una de ellas actúa como un «contenedor», con sus propias librerías y servicios, pero utilizando el SO subyacente.

Los nombres dados a las instancias son el de *containers*, *virtualization engines* (VE) o *jails* (por ejemplo, *FreeBSD jail* o *chroot jail*). A ojos de los usuarios, este «contenedor» es similar a un servidor real y tendrá todos los elementos necesarios, como si del servidor real/virtualizado se tratara, pero con solo un pequeño gasto de memoria, CPU y disco. Una de las desventajas de los contenedores es que, al compartir el SO, no se pueden tener contenedores con sistemas operativos que no compartan el mismo núcleo (por ejemplo, si el *host* es Linux, no podremos poner un contenedor con Windows –cosa que sí es posible si fuera una máquina virtual–).

Software de virtualización del SO de código abierto

Entre los software de virtualización del SO de código abierto, podemos citar Docker (https://www.docker.com), LXC/LXD (https://linuxcontainers.org), OpenVZ (https://openvz.org) y FreeBSD Jails (https://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/jails.html); o en software propietario: Virtuozzo (basado en OpenVZ) (https://virtuozzo.com).

La computación en la nube también comparte características con otros modelos o paradigmas de cómputo distribuido (algunos de futuro) como los siguientes:

1) ***Dew computing***: nuevo paradigma de cómputo distribuido que considera que los **equipos locales** (ordenadores de sobremesa, portátiles y dispositivos móviles) proporcionan un entorno propicio para microservicios independientes de los servicios en la nube y que estos pueden colaborar con los servicios. Es decir, este paradigma plantea la distribución de las cargas de trabajo entre servidores de la nube y los dispositivos discretos/locales para utilizar plenamente el potencial de ambos. Algunos autores hablan más de *mobile cloud computing*, en el que se integran y distribuyen los servicios y el almacenamiento de los datos en los dispositivos móviles y los proveedores de servicios en la nube (algunas experiencias acercan más este paradigma al *volunteer computing*).

2) ***Edge computing***: es un paradigma de computación distribuida que proporciona servicios de datos, computación, almacenamiento y aplicaciones más cerca de los dispositivos cliente o en los dispositivos cercanos al usuario. Es decir, procesa y almacena los datos sobre los dispositivos en el borde de la red (por ejemplo, dispositivos móviles) en vez de enviar los datos a un lugar en la nube. Es considerado una forma de computación distribuida **de proximidad**, en la que cada uno de los dispositivos conectados a la red puede procesar los datos y transmitir solo un conjunto de ellos que sean de interés o hacerlo solo en situaciones excepcionales (alarmas o notificaciones), con capacidad autónoma y sin dependencia del servidor en la nube. Es un modelo que, con el crecimiento que se espera de la IoT (*Internet of things*), permitirá que los sistemas en la nube y las redes no se colapsen ante el incremento exponencial de datos que transmitir, almacenar y procesar. Algunos autores lo diferencian del *fog computing*, porque en este se utilizan agrupaciones/multitudes de usuarios finales (o dispositivos cercanos al usuario) para almacenar datos (en lugar de hacerlo sobre la nube), y por la comunicación (en lugar hacerlo a través de Internet), el control, la configuración, la medición y la gestión (en lugar de ser controlado principalmente por *gateways* de red, como ocurre en la red LTE, por ejemplo).

3) ***Peer-to-peer computing***: este paradigma plantea el uso de una arquitectura distribuida sin la necesidad de una coordinación central para realizar cómputo y almacenamiento de datos. Los participantes son los **proveedores y consumidores** de recursos (en contraste con el modelo tradicional de cliente-servidor) y todos se ayudan para procesar, almacenar y comunicar los datos de forma igualitaria. Los *peer* ponen una parte de sus recursos (potencia de procesamiento, almacenamiento o ancho de banda de red) directamente a disposición de otros *peers* de la red, sin la necesidad de una coordinación central por los servidores. Este tipo de paradigma ha tenido mucha aceptación en servicios de gestión de contenidos (Bittorrent, Spotify), compartición de archivos (Gnutella, Edonkey), dinero digital (Bitcoin, Peercoin) y anonimización (I2P), entre otros.

Lecturas complementarias

David Edward Fisher; Shuhui Yang (2016). «Doing More with the Dew: A New Approach to Cloud-Dew Architecture» [en línea]. *Open Journal of Cloud Computing* (vol. 3, núm. 1). <https://www.ronpub.com/publications /OJCC_2016v3i1n02_Fisher.pdf>

Atta ur Rehman Khan; Mazliza Othman; Sajjad Ahmad Madani; Samee Ullah Khan (2014). «A Survey of Mobile Cloud Computing Application Models» [en línea]. *IEEE Communications Surveys & Tutorials* (vol. 16, núm. 1). <http://ieeexplore.ieee.org/document/ 6553297>

Pedro García López y otros (2015, octubre). «Edge-centric Computing: Vision and Challenges» [en línea]. *ACM SIGCOMM Computer Communication Review* (vol. 45, núm. 5, págs. 37-42). <https://doi.org/10.1145/2831347.2831354>

Jeff Burt (2014, 29 de enero). «Cisco Moving Apps to the Network Edge for Internet of Things» [en línea]. *eWeek*. <https://www.eweek.com/networking/ cisco-moving-apps-to-the-network-edge-for-internet-of-things>

Rüdiger Schollmeier (2001). «A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications» [en línea]. *Proceedings First International Conference on Peer-to-Peer Computing* (págs. 0101). <https://doi.org/10.1109/P2P.2001.990434>

4) *Volunteer computing*: como se mencionó anteriormente cuando hablábamos de los superordenadores distribuidos, este tipo de procesamiento distribuido se basa en que los **usuarios** aportan sus recursos (básicamente, procesamiento y almacenamiento) para un determinado proyecto del cual forman parte. El primer registro que se tiene de este tipo de cómputo distribuido es de 1996, el *Great Internet Mersenne Prime Search* (búsqueda de número primos que cumplen con $2^n - 1$, por ejemplo, 3, 7, 31, etc.), seguido por distributed.net (en 1997). Luego vinieron muchos otros, entre ellos Bayanihan (http://groups.csail.mit.edu/cag/bayanihan), cuyos desarrolladores fueron los que propusieron el nombre de *volunteer computing*. En la actualidad, muchos de ellos están basados en la arquitectura Boinc (desarrollada por la Universidad de California en Berkeley), con proyectos como los ya mencionados anteriormente (Seti@home) o aquellos que han desarrollado su propia arquitectura software, como Folding@home (Universidad de Stanford), que en mayo de 2016 anunciaron que habían llegado a la marca de 100 petaflops (×86).

Lectura complementaria

Luis F. G. Sarmenta (2001). *Volunteer computing* [en línea]. Cambridge: Massachusetts Institute of Technology. <http://people.csail.mit.edu/lfgs/papers/sarmenta-phd-mit2001.pdf>

3. Computación en la nube

Como complemento del punto anterior, y dado que ya hemos hablado de *nube* (otra palabra también de moda), se ampliará este concepto para establecer algunas ideas y referencias en cuanto a qué significa la computación en la nube (*cloud computing*), las ventajas y desventajas de esta forma de disponer y gestionar recursos que ya forma parte de nuestras vidas (redes sociales, cuentas de correo, aplicaciones ofimáticas, etc.), y que solo con un navegador (o una aplicación de un móvil) podemos procesar nuestros datos y gestionar nuestra información de forma remota.

El *cloud computing*[1] es una propuesta tecnológica adoptada por la sociedad en general como forma de interacción entre proveedores de servicios, gestores, empresas/administración y usuarios finales para la prestación de servicios y la utilización de recursos en el ámbito de las tecnologías de la información y la comunicación (TIC), y sustentado por un modelo de negocio viable económicamente.

(1)Conocido en castellano como computación en la nube, servicios en la nube, informática en la nube, nube de cómputo o nube de conceptos.

Una definición interesante que aporta más definición sobre la computación en la nube es la del National Institute of Standards & Technology (NIST):

> «El *cloud computing* es un modelo que permite acceso ubicuo, a conveniencia, bajo demanda y por red a un conjunto compartido de recursos informáticos configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente aprovisionados y liberados con el mínimo esfuerzo de administración o sin interacción con el proveedor de servicios».

Este paradigma, incuestionable en cuanto a su aceptación actual, vincula a proveedores, usuarios y a toda su cadena intermedia de transformación y procesamiento de la información a través de una red que, en la mayoría de los casos, es Internet. Es habitual que los usuarios utilicen servicios en la nube (ya sean servicios empresariales o personales), como el correo electrónico, las redes sociales o el almacenamiento de archivos (fotos, vídeos, ofimática), y que los utilicen desde varios dispositivos y con distintas interfaces. Tan común es este paradigma, que los usuarios digitales (jóvenes generaciones) no conocen ya otra forma de hacerlo: por un lado, han nacido con esta tecnología que, junto con la del móvil, han transformado la sociedad actual y, por otro, la interconectividad o usabilidad de diferentes interfaces/medios de acceso son esenciales para su quehacer cotidiano (compras, ocio, servicios administrativos, negocio y una larga lista más).

3.1. Breve historia

Sin olvidar las grandes compañías precursoras que ofrecieron este tipo de servicios (Yahoo! en 1994 y Google en 1998 –motor de búsqueda–, Hotmail en 1997 –correo–, Amazon en 2002 y AWS en 2006), el concepto hunde sus raíces en los años sesenta y en las ideas de John McCarthy (profesor emérito de la Universidad de Stanford y cofundador del Laboratorio de Inteligencia Artificial del MIT), quien, en 1961, durante un discurso para el centenario del MIT, predijo que el **uso a tiempo compartido** (que estaba en auge en aquel momento) de los ordenadores podría conducir a un futuro en el que el poder del cómputo, e incluso de aplicaciones específicas, podría ser vendido como un **servicio**.

Esta idea, aunque desde otro punto de vista, también fue expresada por otros investigadores, especialmente Joseph Carl Robnett Licklider (conocido como JCR o simplemente Lick), una de las figuras más importantes en el ámbito de la ciencia computacional, recordado particularmente por ser uno de los primeros que imaginó la computación interactiva moderna y su aplicación a diferentes actividades con una visión de la interconexión global de ordenadores (mucho antes de que fuera implementada). JCR trabajó y aportó financiación a proyectos como Arpanet (predecesor de Internet) o la interfaz gráfica de usuario, que permitieron el acceso a servicios y recursos a personas no especialistas, ideas que, en opinión de expertos actuales, se asemejan mucho a la computación en la nube tal como la conocemos hoy.

Pero la historia de la computación en la nube también se sustenta en visionarios como John Burdette Gage. Durante su estancia laboral en Sun Microsystems, Burdette vaticinó que «***the network is the computer***», y más recientemente, George Gilder (cofundador del Discovery Institute) escribió en 2006 el artículo «The information Factories», en el que afirmaba que el PC ya no era el centro de atención y almacenamiento/procesamiento de la información, lugar que había cedido a la nube, que sería la responsable de almacenar los datos de cada individuo en el planeta y al que accederían por lo menos una vez en la vida. En 2013, Gilder publicó *Knowledge and Power: The Information Theory of Capitalism and How It is Revolutionizing Our World*, obra en la que relacionaba la economía, el capitalismo y la teoría de la información de Alan Turing y Claude Shannon, y cómo afectarían a la sociedad.

Las evoluciones desde esos años sesenta inicales han sido notables e intrépidas, pero probablemente la más destacable en los aspectos más vinculados a la nube actual sea el **desarrollo de la Web**[2] y su evolución más reciente a la Web 2.0 y el desarrollo de Internet (que en España no tuvo un impacto sustancial hasta principios de los años noventa con la transformación de RedIris y la conexión plena a Internet).

(2)Tim Berners-Lee propone las ideas en 1989, pero no fue hasta 1990 cuando se creó un prototipo de la *World Wide Web*.

A partir de entonces, comienzan a surgir una serie de **servicios**, básicamente aplicaciones centradas en la búsqueda y el correo, en páginas como Wandex (1993, que pretendía ser un programa para medir el tamaño de Internet desarrollado por el MIT, pero que acabó siendo el primer buscador), Yahoo! (1994), Altavista (1995, que ha ido cambiando de propietarios, desde Overture, Yahoo y ahora Microsoft), Hotmail (1996, posteriormente comprado por Microsoft en 1997, cuando ya contaba con seis millones de usuarios de correo) o más recientemente el todopoderoso Google (1998).

Para algunos autores, uno de los grandes hitos vinculados a la nube es la llegada, en 1999, de **Salesforce.com**,[3] una de las primeras empresas en ofrecer a sus clientes lo que hoy se llama SaaS (*software as a service*) a partir de un software para la automatización de ventas mediante una simple página web. Algunos expertos consideran que este servicio actualmente puede ser comparado con un PaaS (*platform as a service*) y otros lo consideran una plataforma híbrida que ya ofrece las dos variantes. Esto permitió mostrar una «nueva» forma de negocio a través de Internet y generar lo que hoy es una realidad a diferentes niveles de aplicaciones, servicios y recursos en la red, conocida como computación en la nube (*cloud computing*).

(3)Empresa que ha sido considerada la más innovadora en los últimos cuatro años en el ranking elaborado por Forbes.

En 2002, Amazon anunció su infraestructura en la red y su ampliación en 2006 con el lanzamiento de ***Amazon Web Service*** **(AWS)**, que incorporaba *elastic compute* (llamado comúnmente EC2) y almacenamiento en la nube (*storage cloud* o S3), los servicios más importantes de AWS para proveer instancias de un servidor bajo demanda para cómputo y almacenamiento, respectivamente. Estos servicios orientados a negocio permitieron a las pequeñas empresas y particulares usar equipos en los que ejecutar sus propias aplicaciones y con un modelo de monetización basado en el «pago por uso». En 2009, Google y otros grandes proveedores empezaron a ofrecer aplicaciones basadas en navegador, que se hicieron muy populares y ganaron en seguridad y servicio. También otros grandes proveedores salieron al mercado con sus propias infraestructuras en la nube.

Proveedores con infraestructuras propias en la nube

Microsoft Azure fue anunciado en octubre del 2008 (aunque no empezó a prestar servicios hasta 2010), IBM en 2011 (lanzó SmartCloud Framework para dar soporte a su proyecto *Smarter Planet*) u Oracle en 2012 (Oracle Cloud).

En cuanto a las plataformas más representativas (no las únicas) para construir *clouds*, destacamos las siguientes:

- En 2005 se comenzó a trabajar en un proyecto de investigación orientado a este ámbito basado en software de código abierto: **OpenNebula** (http://opennebula.org), que vio la luz en 2008. Ese año se formó el consorcio OpenNebula.org, financiado por un proyecto de la UE (7th FP), que en 2010 llegó a tener 16.000 máquinas virtuales (MV) configuradas.

- En 2010, Rackspace y la NASA lanzaron una iniciativa de software en la nube de código abierto, **OpenStack** (https://www.openstack.org), que nació con el objetivo de ofrecer a las organizaciones la posibilidad de montar sus servicios de nube sobre hardware estándar (en 2012 se creó la OpenStack Foundation para promover este software en la comunidad, con 200 socios, gran parte de todos los actores importantes del mundo TIC).

- También en 2010, una empresa llamada Cloud.com liberó su producto orientado a la creación de *clouds*, llamado **CloudStack** (sobre el que había estado trabajando en secreto durante los años anteriores), bajo GPLv3. En 2011, Cloud.com fue adquirida por Citrix Systems y CloudStack pasó a ser un proyecto de la Fundación Apache y distribuirse bajo la *Apache Software License* (https://cloudstack.apache.org).

A modo de resumen, podemos decir que en la actualidad es poco frecuente que una institución o una empresa mediana-grande no tenga externalizado algunos servicios en la nube para sus trabajadores/usuarios finales o no utilice la nube para algunos de los aspectos vinculados al negocio de la misma (correo, web, intranet, etc.), excepto en aquellas cuyo negocio principal dependa de la privacidad de los datos o estos estén especialmente protegidos.

3.2. Características, ventajas y desventajas

Si tenemos en cuenta las opiniones de los expertos (por ejemplo, Jamie Turner), además del Web 2.0, las grandes revoluciones que han facilitado la evolución de la computación en la nube son el avance de las tecnologías de virtualización (técnica que permite poner más de un ordenador con su sistema operativo ejecutándose en el mismo hardware y compartir recursos), la proliferación a un coste aceptable del ancho de banda y los estándares de interoperabilidad de software para facilitar que cualquier recurso o servicio pueda hoy en día tener como base la nube.

No obstante, estas opiniones tan favorables hacia la nube deben considerarse en un marco adecuado, y no caer en la tentación de creer que la nube lo puede contener y proveer todo. Como todo sistema, tiene sus elementos positivos (básicamente, acceso a recursos y servicios por parte de los usuarios sin ser expertos para su instalación y administración, flexibilidad de uso y adaptativa, y precios aceptables), pero también sus puntos débiles, que conviene tener muy presentes, especialmente su talón de Aquiles: la disponibilidad 24/7 del acceso a la red y su ancho de banda.

Sin red o con una red deficiente, la nube no existe.

Las características clave para la computación en la nube son las siguientes (en orden alfabético):

1) Agilidad y autoservicio: rapidez y capacidad de proveer recursos (de forma casi inmediata) sin grandes intervenciones ni acciones por parte de las dos partes. Es decir, el usuario puede, de forma no asistida, aprovisionar capacidades de cómputo/almacenamiento/redes, según necesite, automáticamente, sin necesidad de interacción humana con el proveedor de servicios. Estos recursos estarán disponibles en un intervalo corto de tiempo (segundos o a lo sumo unos pocos minutos) a través de la red.

2) Coste: dada la eficiencia y la automatización en la gestión y la administración de recursos, los precios para los proveedores de en la nube son reducidos y pueden trasladarlos al usuario final, de manera que este no debe hacer frente a los gastos derivados de la implantación de infraestructura civil y los servicios, y a la inversión en máquinas, software y recursos humanos, lo que hace que todo el modelo sea muy favorable. Además, como el control y la monitorización son automáticos, se pueden usar medidas con un cierto nivel de abstracción, muy apropiadas para este tipo de servicio (por ejemplo, cuentas/cuentas activas, almacenamiento, procesamiento o ancho de banda), de manera que resulta posible ajustar el coste a modelos de pago por uso, pago por peticiones, etc., y se proporciona transparencia tanto para el proveedor como para el consumidor del servicio utilizado.

3) Escalabilidad y elasticidad: estos conceptos se refieren al aprovisionamiento de recursos en (casi) tiempo real y la adaptabilidad a las necesidades de carga y uso de los mismos. Es decir, si podemos prever la carga y la utilización, no es necesario aprovisionar todos los recursos desde el inicio, como si de una inversión local se tratara, sino planificarlos para cuando la demanda lo requiera, con la consiguiente reducción del coste.

4) Independencia entre el dispositivo y la ubicación: independizar el recurso del acceso y facilitar que los recursos puedan usarse desde una red local, corporativa u otro tipo y desde diferentes dispositivos (capacidad/tipología).

5) Mantenimiento y licencias: el modelo de actualizaciones y licencias se simplifica, ya que el software se encontrará en el servidor de la nube y no habrá nada instalado en el dispositivo del usuario final.

6) Rendimiento y gestión de recursos: control exhaustivo y monitorización eficiente de los servicios para lograr una alta disponibilidad y una utilización óptima de los recursos de forma automática. Estas características permiten reducir al máximo las ineficiencias y aportan control, notificación inmediata, transparencia y seguimiento tanto al proveedor como al usuario final.

7) Seguridad: puede incrementarse porque los datos están centralizados. La seguridad de la aplicación le corresponde a su responsable, mientras que al proveedor le corresponde la responsabilidad de la seguridad física. Es decir, la seguridad será como mínimo la misma que la de los sistemas tradicionales, pero podrá ser mejorada y deberá estipularse qué nivel se desea en el contrato de prestación de servicios, el SLA (*service level agreement*).

8) Virtualización como base para el aprovisionamiento: esto permite compartir y optimizar el uso del hardware reduciendo los costes, la energía consumida y el espacio, lo que facilita el despliegue rápido de servicios y garantiza la alta disponibilidad (servicios en espera) y la movilidad por carga (movimiento de máquinas virtuales a servidores más ociosos cuando la carga aumenta).

Para convencer a los más reticentes, podemos enumerar un conjunto de ventajas que aporta la nube como paradigma (en orden alfabético):

1) Fácil integración y aceptación: gracias a su desarrollo y a estar basado en estándares, se puede integrar con mucha mayor facilidad y rapidez con el resto de las aplicaciones empresariales. El usuario final queda totalmente convencido cuando se le permite acceder desde cualquier dispositivo y sin requerimientos especiales en cuanto a las necesidades del mismo.

2) Servicio global: ubicuidad y acceso a los servicios desde cualquier lugar, con tiempos de inactividad reducidos al mínimo y con alta disponibilidad de recursos.

3) Simplicidad: roles y responsabilidades muy bien definidos, lo que conlleva una separación de las actividades bien estipulada (por ejemplo, proveedor de contenidos, proveedor de aplicación, proveedor de infraestructura y usuario final). Todo ello se traduce en una forma óptima de trabajar y menor inversión para todas las partes.

4) Reducción de riesgos y rapidez: no es necesaria una gran inversión ni la adecuación de sitios y entornos y, además, es posible acelerar al máximo la implantación de nuevos servicios en las diferentes etapas de desarrollo, hasta producción.

5) Reducción del uso de energía (consumo eficiente): dada la tecnología utilizada y la eficiencia en el uso de los recursos (favorecido por la virtualización), solo se consume lo necesario, a diferencia de los centros de datos tradicionales, en los que existe un consumo fijo no dependiente de la carga y la utilización.

Aunque las ventajas son evidentes y muchos usuarios de esta tecnología la usan principalmente por el abaratamiento de los costes de servicio, comienzan a existir opiniones contrarias que consideran que una nube pública puede no ser la opción más adecuada para determinado tipo de servicio o infraestructura. Algunas de las principales desventajas que esgrimen los expertos son las siguientes (por orden alfabético):

Lectura complementaria

Paul Haefele (2012). «EC2 is 380% more expensive than internal cluster» [en línea]. *Deep Value*. <http://deepvalue.net/ec2-is-380-more-expensive-than-internal-cluster>

1) Centralización: tanto de los datos como de las aplicaciones, lo cual genera una dependencia del proveedor. Si este no dispone de la tecnología adecuada (monitorización y detección) ni los recursos apropiados (alta disponibilidad), se pueden generar cortes o inestabilidades en el servicio. En estos casos, adquiere una especial relevancia el SLA (*service level agreement*), que especificará a qué está obligado el proveedor y las indemnizaciones a las que deberá hacer frente por ello.

2) Confiabilidad: la «salud» tecnológica y financiera del proveedor será un elemento clave en la continuidad de su negocio, y también del nuestro, por lo que las decisiones que tome afectarán directamente al negocio cliente y a la empresa. Esta podría quedar a merced de un mercado muy dinámico en cuanto a fusiones y monopolios (o pseudomonopolios), con el consiguiente impacto que esto podría tener en los costes de los servicios.

3) Escalabilidad: mientras más clientes tenga el proveedor, a más usuarios deberá abastecer el hardware, con la consiguiente sobrecarga del sistema; si el proveedor no dispone de un plan de escalabilidad a medio y largo plazo, de forma que pueda asegurar un crecimiento sostenible desde el punto de vista de las necesidades de sus clientes, se puede llegar a la saturación de los servicios, con la posterior degradación y pérdida de prestaciones.

4) Especialización o cualificación: la necesidad de servicios «especiales» o cualificados podría tener una prioridad muy baja (y el consiguiente retardo en su implementación) para el proveedor si la demanda es baja.

5) Disponibilidad: el principal punto débil de una infraestructura en la nube es el acceso a Internet. Si no se dispone de un acceso confiable y con un ancho de banda aceptable, la nube deja de ser efectiva.

6) Riesgo y privacidad: los datos «sensibles» de negocio no residen en las instalaciones de las empresas, por lo que su seguridad no depende de los recursos humanos de estas, sino del proveedor del servicio. Si se asume que los datos son de alto valor en un contexto vulnerable, el riesgo puede ser muy alto por su posible robo (copia), acceso a la información (lectura) o destrucción (borrado).

7) Seguridad: el hecho de que la información deba atravesar diferentes canales y servicios hace que cada uno de ellos pueda resultar un foco de inseguridad. Si bien esto puede ser resuelto mediante canales y servicios seguros, la posibilidad

de un fallo en la cadena de cifrado de la información aumenta. Además, es fácil que el propietario de la información desconozca por completo qué ha pasado y dónde se ha producido el fallo.

8) Dependencia de un proveedor (*vendor lock-in*): es uno de los grandes problemas (que en la actualidad se ha demostrado como uno de los más frecuentes). Al depender de un proveedor de productos y servicios contractualmente, muchas veces los clientes no pueden usar los servicios de otros proveedores por la penalización de los elevados costes que implicaría el cambio, aunque esto suponga una atrayente reducción de sus costes o el acceso a mejores prestaciones. Muchos desarrolladores o usuarios finales son reticentes a la nube sobre todo por este motivo.

Como se puede ver entre características, ventajas y desventajas, pueden existir elementos que entran en contradicción (por ejemplo, el tema de la seguridad). Así, pues, el equipo encargado de tomar las decisiones de la empresa deberá sopesar cuidadosamente las distintas posibilidades, con sus ventajas y sus riesgos.

Desde un punto de vista global, la computación en la nube es una tecnología que aporta beneficios: tras una adecuada planificación, después de valorar bien todos los factores que influyen en el negocio y dejando de lado los conceptos superficiales (todos lo tienen, todos lo utilizan, bajo coste, expansión ilimitada, etc.), puede ser una elección adecuada para los objetivos de la empresa, para su negocio y la prestación de servicios de TIC que necesita.

La figura 2 muestra una visión general de los actores y elementos en juego en la computación en la nube:

- **Infraestructura**: recursos de hardware y software que gestiona el proveedor y que serán usados por las empresas y los clientes.

- **Catálogo de servicios**: elementos ofrecidos por la nube, que serán seleccionados por los clientes del proveedor de servicios por categorías o prestaciones.

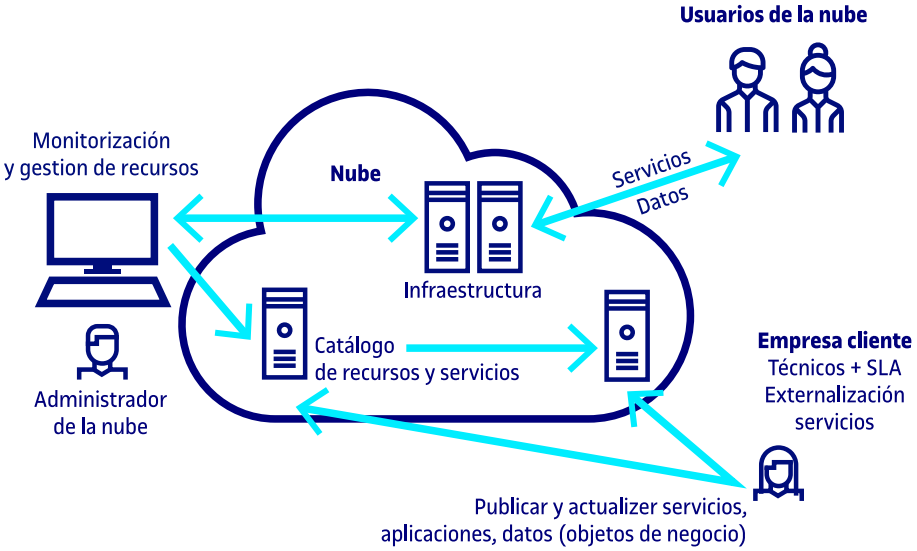
- **Administrador de la nube**: cuerpo técnico de profesionales que garantizarán las prestaciones, la seguridad, la estabilidad, la disponibilidad, etc. de los servicios comercializados a partir de un SLA firmado con cada cliente.

- **Empresa cliente**: usuarios que realizan una externalización de sus servicios de TIC, y publican y actualizan sus aplicaciones/datos en la nube. Se basan en una garantía del servicio que ofrece el proveedor, que forma parte

de un acuerdo de prestación de servicios (SLA), cuyo coste estará basado generalmente en políticas similares al pago por uso.

- **Usuarios**: clientes de la empresa que ha puesto sus servicios en la nube, que puede conocer o no que los mismos están siendo provistos desde la nube. Solo acceden a un servicio, una aplicación o datos, como si entraran en los servidores de la empresa que provee el servicio.

Figura 2. Actores y elementos en juego en la computación en la nube



Es importante destacar que los usuarios (a veces denominados *usuarios finales*) generalmente no son clientes del proveedor de servicios de la nube, sino de la empresa que provee los servicios en la nube.

Clientes de Amazon AWS

Entre los principales clientes de Amazon AWS (es decir, hablamos de empresas que tienen la etiqueta «millones de dólares» asociada a alguna característica –ingresos, activos totales, financiación, valoración o beneficios–), encontramos Adobe Systems, Airbnb, Alcatel-Lucent, Aon, Autodesk, BMW, Bristol-Myers, Canon, Capital One, Comcast o Docker, entre otras muchas. Para Adobe Systems, por ejemplo, el proveedor de la nube y de la infraestructura es AWS, y la empresa que contrata es Adobe, que comercializa los productos a través de Creative Cloud –diferente software de diseño gráfico, edición de vídeo, diseño web y servicios en la nube–, que llegan a los usuarios (finales) por una cuota económica mensual.

3.3. Clasificación

Existen diferentes taxonomías para describir los servicios y la forma de despliegue de la nube. Como modelos de despliegue, podemos enumerar los siguientes:

Lectura recomendada

En el siguiente documento se definen cuatro modelos de despliegue y tres de servicio en la nube:

Peter Mell; Timothy Grance (2011, septiembre). «The NIST Definition of Cloud Computing» [en línea]. *National Institute of Standards and Technology*. <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800 -145.pdf>

1) Nube pública: el sistema es abierto para uso general, bajo diferentes modelos de negocio (desde pago por recursos, por uso, cuota o libre). El recurso aquí es mantenido y gestionado por un tercero, y los datos y aplicaciones de los diferentes clientes comparten los servidores, los sistemas de almacenamiento, las redes y otras infraestructuras comunes. Normalmente los recursos se utilizan y gestionan a través de Internet. Generalmente, los clientes no tienen conocimiento de con quién comparten la infraestructura. El uso se contrata a través de un catálogo de recursos disponibles y su aprovisionamiento es automático (o semiautomático), después de haber cumplido con una serie de trámites en cuanto al modelo de pago y SLA.

El propietario de la nube (proveedor de recursos y servicios) puede ser una empresa privada, una institución académica, una organización gubernamental o una combinación de estas, generalmente en función del tipo de clientes y de los recursos que se tienen que proveer. Técnicamente puede haber pocas diferencias (o ninguna) con alguno de los otros modelos (especialmente con el privado); sin embargo, puede haber diferencias sustanciales en cuanto a la seguridad: la nube pública puede estar disponible en una red abierta como Internet y sin mecanismos de cifrado.

Nube pública

Como ejemplo de servicios de nube pública, podemos mencionar AWS (Amazon Web Services), Microsoft Azure, Google Compute, Rackspace, AliBaba Cloud o IBM-SoftLayer, entre otros, que operan su propia infraestructura y sus recursos son accesibles por Internet; también el CSUC (Consorci de Serveis Universitaris de Catalunya), que ofrece, bajo diferentes modelos de explotación, recursos de nube para instituciones académicas y de investigación de Cataluña (https://www.csuc.cat/es/servicios/servicios-en-la-nube).

2) Nube privada: en este caso, la infraestructura funciona exclusivamente para una organización/empresa y es utilizada por sus unidades de negocio o departamentos. La empresa puede ser propietaria, administradora y operadora, pero alguno de estos elementos también puede ser subcontratado a un tercero. Es la opción más favorable para las compañías que necesitan alta protección de datos y un alto nivel de servicio, ya que los recursos y su gestión se encuentran bajo el control y cuidado de la propia empresa/organización.

Aunque este modelo resuelve algunos problemas de cierto tipo de empresas (por ejemplo, las que cuentan con una legislación especial, como las empresas públicas), estas deben asumir el coste de la inversión (CAPEX). Como contrapartida, disponen de una infraestructura bajo demanda gestionada por personal propio (o externo, pero bajo sus criterios), que controla qué aplicaciones usar y dónde deben ejecutarse, de forma que mantienen la privacidad de su

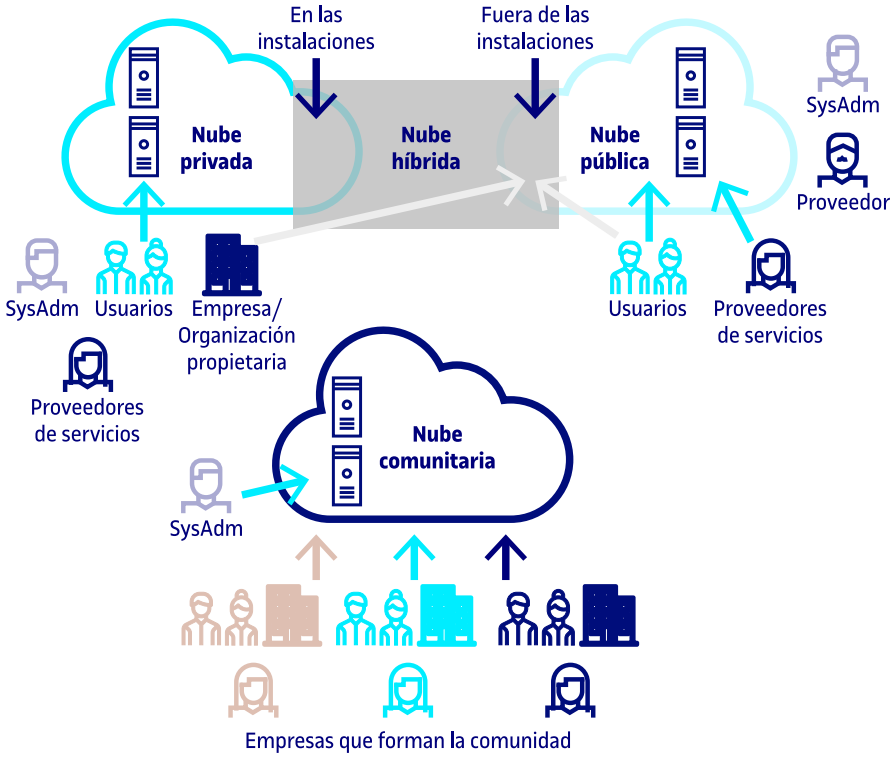
información, lo que permite definir las políticas de acceso y evitar el *lock-in* de los datos, así como la dependencia de un proveedor mencionada anteriormente.

3) Nube híbrida: se combinan modelos públicos y privados, de manera que el propietario dispone de una parte privada (con acceso controlado y bajo su control) y otra compartida, aunque de una manera controlada. Las nubes híbridas ofrecen la posibilidad de escalar muy rápidamente, con aprovisionamiento externo bajo demanda, pero implican una gran complejidad a la hora de decidir cómo se distribuyen los datos y las aplicaciones en una parte u otra. Aunque se trata de una propuesta atractiva para las empresas, habitualmente su uso no pasa de aplicaciones simples, sin restricciones de sincronización y con bases de datos sofisticadas (por ejemplo, las implementaciones de correo empresarial o aplicaciones de ofimática).

4) Nube comunitaria: los servicios están diseñados para que puedan ser utilizados por una comunidad, organizaciones o empresas bajo el mismo alineamiento de objetivos o negocio (por ejemplo, bancos, distribuidores, arquitectos, etc.) o que requieran unas características específicas (por ejemplo, seguridad y privacidad). Las empresas que forman parte de la comunidad pueden ser propietarias de la infraestructura, así como gestores u operadores; algunos de estos roles también pueden ser subcontratados a terceros, pero bajo las indicaciones y reglas que aplica la comunidad.

La figura 3 muestra un esquema de estas cuatro formas de implantación y desarrollo de las nubes.

Figura 3. Formas de implantación y desarrollo de las nubes



Otra de las clasificaciones habituales es a partir del **nivel de servicio** que prestan. En este caso, las tres tradicionales son las siguientes:

1) IaaS (*infraestructure as a service*).
2) SaaS (*software as a service*).
3) PaaS (*platform as a service*).

Pero, en la actualidad, se pueden encontrar otras extensiones o derivadas, como pueden ser BaaS (*business as a service*), StaaS (*storage as a service*), DaaS (*desktop as a service*), DRaaS (*disaster recovery as a service*), MaaS (*marketing as a service*). Algunos autores les dan un nombre global: XaaS (*everything as a service*) para referirse a la creciente diversidad de servicios disponibles en la nube a través de Internet. Un caso que afirma estas consideraciones es el de aPaaS (*application platform as a service*), que permite un rápido desarrollo y aprovisionamiento de aplicaciones como plataforma específica para la codificación, aprovisionamiento y despliegue de aplicaciones, y que soporta el ciclo de vida completo proporcionando una forma más rápida de crear aplicaciones.

En la **infraestructura como servicio (IaaS)**, se encuentra la capa de recursos básica (generalmente máquinas virtuales, redes y almacenamiento) en la que el cliente podrá instalar sus SO y sus aplicaciones, para implementar un servicio o ejecutar las aplicaciones. El cliente no podrá manejar o controlar el hardware subyacente, pero sí el SO, el almacenamiento y las aplicaciones y servicios implementados sobre esas máquinas, así como algunos aspectos de la conectividad (subredes, cortafuegos, dominios, etc.).

Proveedores de IaaS

Ejemplos de grandes proveedores de IaaS (decenas a miles de MV) son Amazon EC2 (https://aws.amazon.com/free), Google Compute Engine (https://console.cloud.google.com/freetrial) o Digital Ocean (https://www.digitalocean.com) como servicio representativo para PYMES (en unidades de MV).

La reflexión del usuario en esta modalidad sería: «**¿Por qué comprar, instalar y probar infraestructura cada *N* años (obsolescencia) y no alquilarla y pagar por uso?**». Uno de los principales atractivos de la IaaS es que los recursos están disponibles sin hacer obra civil (centro de datos), con mínimos recursos humanos (no especialistas en TIC), sin gran inversión inicial, escalable, eficiente y a un coste aceptable.

La **plataforma como servicio (PaaS)** es la encapsulación de un ambiente de desarrollo y la provisión de una serie de módulos que proporcionan una funcionalidad horizontal (persistencia de datos, autenticación, mensajería, etc.). Una estructura básica de este tipo podría consistir en un entorno que contenga servicios/aplicaciones, librerías y API para un fin específico (por ejemplo, para una tecnología de desarrollo en particular sobre Linux y un servidor web + una base de datos y un ambiente de programación como Perl o Ruby). Es decir, el cliente no gestiona ni controla la infraestructura (servidores, sistemas operativos, almacenamiento, ni ningún tipo de elementos de red o seguridad, etc.), pero tiene control de las aplicaciones desplegadas y de la configuración del entorno de ejecución de las mismas (bases de datos y *middlewares*). Es habitual que una PaaS pueda prestar servicios en todos los aspectos del ciclo de desarrollo y pruebas de software, o en el desarrollo, la gestión y la publicación de contenidos.

Proveedores de PaaS

Algunos ejemplos de PaaS son Codeanywhere (https://codeanywhere.com), Google App Engine (https://cloud.google.com/products), Heroku (https://www.heroku.com) u OpenShift (https://www.openshift.com).

Podríamos decir que en esta modalidad, el deseo del usuario sería algo así como: «**Necesito el software OPQ instalado sobre el sistema operativo RST, y la base de datos UVW con la API XYZ**», por lo que recibiría todo el conjunto (HW, SO, base de datos, librerías, API, seguridad, GUI y otras herramientas) listo para usar en pocos segundos para desarrollar aplicaciones empresariales/móviles, páginas web, contenidos, etc.

Finalmente, el **software como servicio (SaaS)** se encuentra en la capa abstracta (si bien existe una tendencia bastante extendida a considerar que el SaaS es la capa más simple del PaaS, o la más baja) y caracteriza una aplicación completa ofrecida como un servicio. En general, las aplicaciones bajo este modelo de servicio son accesibles a través de un navegador web y el usuario no tiene control sobre ellas, aunque en algunos casos se le permite realizar algunas configuraciones. Así, el cliente no necesita instalar la aplicación en sus propios

ordenadores, de forma que elimina el soporte y mantenimiento del hardware y del software, además de mejorar el control y la gestión de las licencias si son necesarias.

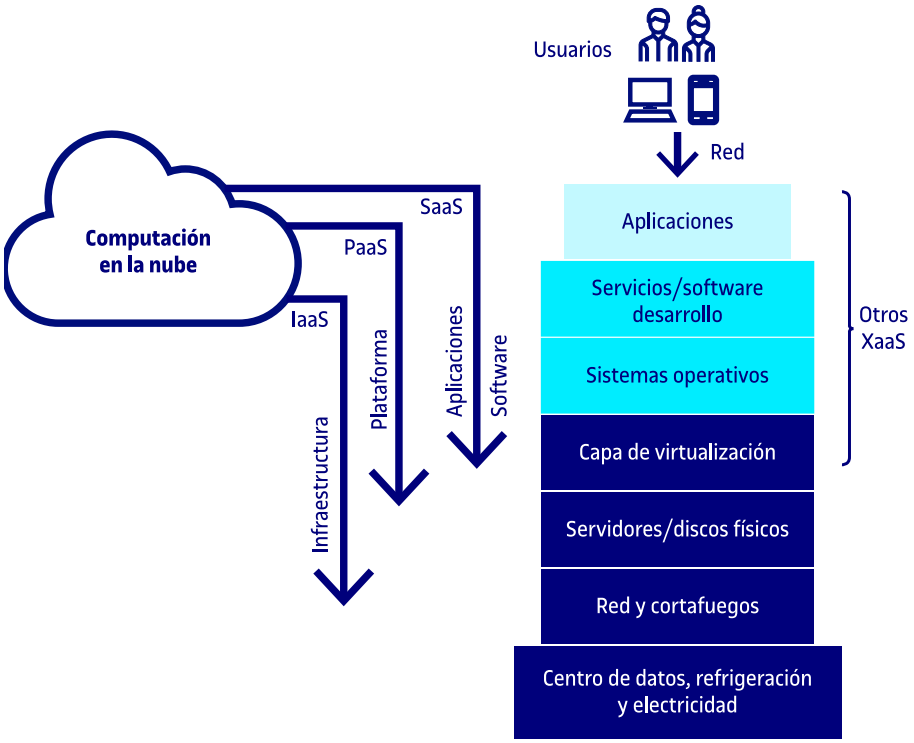
Proveedores de SaaS

Algunos ejemplos representativos de SaaS son WebMail, con Gmail (https://mail.google.com), Outlook (https://outlook.live.com), Yahoo (https://login.yahoo.com), etc.; SalesForce (https://www.salesforce.com/es/products), Google Docs (https://www.google.es/intl/es/docs/about), Office 365 (https://products.office.com/es-es/business/office) o SiteBuilder (https://www.sitebuilder.com).

Aquí el pensamiento del usuario sería algo así como: «**Ejecuta esto por mí**», sin responsabilidades en la gestión del hardware o el software, de manera que utiliza un navegador web o evita la instalación de software cliente, basado en un servicio bajo demanda, con escalabilidad casi inmediata, acceso desde cualquier sitio y con cualquier dispositivo, bajo un modelo de soporte 24/7 y un modelo de pago como el pago por uso (*pay as they go*) y *pay as they grow*).

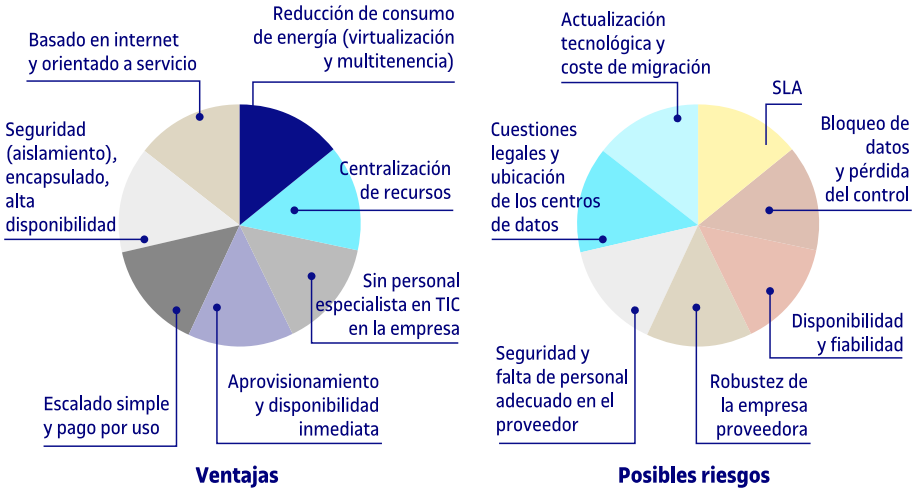
La figura 4 resume las diferentes modalidades de servicio de la nube y qué implican cada una de ellas.

Figura 4. Modalidades de servicio de la nube y sus implicaciones



En la figura 5 se muestran, de forma resumida, tanto las características a favor de la computación en la nube como aquellas que no son tan favorables, que pueden representar un riesgo y se deberán analizar con cuidado.

Figura 5. Ventajas y posibles riegos de la computación en la nube



4. El concepto de *big data*

El *big data* se ha transformado en una palabra de moda. Tanto en centros de R +D+i como en las empresas, las industrias y las instituciones hay cada vez más cantidad de datos y son necesarias nuevas técnicas y herramientas para procesarlos y obtener información de ellos. El Foro Económico Mundial declaró los datos como un **nuevo activo económico**, como la moneda o el oro. La revolución digital y electrónica no solo ha aumentado enormemente el volumen de datos (como se comentó anteriormente), del orden de unos cuantos exabytes diarios, sino también su variedad (es decir, estructurados, semiestructurados y no estructurados) y su velocidad de generación (millones de dispositivos y aplicaciones que generan datos enormes cada segundo).

Lectura complementaria

Foro Económico Mundial (2012). *Big Data, Big Impact: New Possibilities for International Development* [en línea]. <https://www3.weforum.org/docs/WEF_TC_MFS_BigDataBigImpact_Briefing_2012.pdf>

Este enorme crecimiento de los datos ha llegado al **límite del procesamiento y el almacenamiento** de las infraestructuras de gestión de la información existentes, lo que ha obligado a las empresas a invertir en más hardware y actualizaciones de bases de datos. Esta tendencia, que aparentemente soluciona el problema inicial, en realidad no es una solución real, ya que el conjunto de datos sigue creciendo, lo cual conduce a un ciclo de necesitar más hardware para más almacenamiento, pero los datos siguen creciendo, y así indefinidamente. Además, la infraestructura tradicional no es eficiente debido a los altos costes, las limitaciones de escalabilidad (cuando se trata de petabytes) y la incompatibilidad de los sistemas de bases de datos relacionales con datos no estructurados.

Para abordar la enorme cantidad de datos en la web, en 2004 Google presentó un modelo de programación llamado *MapReduce* (MR) con la finalidad de realizar, de forma paralela, tareas de búsqueda sobre sus clústeres de servidores. Para lograr este objetivo, publicaron sus ideas sobre un sistema de archivos distribuidos (para tener los datos cerca de donde se tuvieran que procesar) en 2003 y luego, en diciembre de 2004, el algoritmo de procesamiento **MapReduce**.

Basándose en estas ideas, dos investigadores, Doug Cutting y Mike Cafarella, crearon un sistema de archivos y un entorno de procesamiento, y realizaron las implementaciones pertinentes para ejecutar Nutch (un motor de búsqueda) sobre esta infraestructura. En 2006, con Doug Cutting ya trabajando en Yahoo, hicieron mejoras en el software que habían probado con Nutch y crearon un entorno llamado *Hadoop* (http://hadoop.apache.org) como un proyecto de código abierto en la Apache Software Foundation (https://apache.org). Desde entonces, **Hadoop** se ha convertido en el estándar *de facto* para alma-

cenar, procesar y analizar cientos de terabytes o petabytes de datos, y como un proyecto totalmente de código abierto y disponible para la comunidad que lo desee utilizar incluso con fines comerciales.

Hadoop permite el procesamiento distribuido de una enorme cantidad de datos en un conjunto de servidores de bajo coste, que almacenan y procesan los datos y pueden escalarlos sin límites (este enfoque se denomina *escalado horizontal*).

Por ejemplo, de acuerdo a los datos de 2015, los clústeres de Hadoop en Yahoo! incluían más de 40.000 servidores y almacenaban 40 petabytes de datos de aplicaciones.

Lecturas complementarias

Cade Metz (2011, 18 de octubre). «How Yahoo Spawned Hadoop, the Future of Big Data» [en línea]. *Wired*. <https://www.wired.com/2011/10/how-yahoo-spawned-hadoop>

Sushant Gupta (2020, 2 de septiembre). «What is big data analytics? Beginner guide to the world of big data» [en línea]. *Tricky Enough*. <https://www.trickyenough.com/big-data-analytics>

Después del lanzamiento del entorno Hadoop, la **analítica de *big data*** (*big data analytics*) se transformó en el gran reto tecnológico, que permitiría a las empresas e instituciones dar el salto estratégico de la retrospectiva a la prospectiva, y extraer valor de los grandes conjuntos de datos. Como ocurre con toda nueva idea, muchos escépticos están en contra del «*big data* y lo que comporta». Sostienen que durante décadas las empresas han tomado decisiones comerciales basadas en datos transaccionales almacenados en bases de datos relacionales (DBRMS).

Estas críticas olvidan que existe un enorme potencial en los datos no tradicionales y no estructurados, como los registros web, las redes sociales, el correo electrónico y los datos de sensores, imágenes, audio y vídeo, que pueden contener información útil y valiosa. Estas opiniones tampoco tienen en cuenta lo que ocurre en el mundo real, donde la toma de decisiones basada en datos ha crecido exponencialmente en los últimos años, con una reducción significativa de las decisiones basadas en las evidencias o el instinto/experiencia.

Por otro lado, las ciudades inteligentes (*smart cities*) y el Internet de las cosas (*Internet of things*), precisamente gracias al *big data*, comienzan a ofrecer una **gestión más eficiente** de la energía, el agua y los servicios de transporte en las grandes ciudades, lo que está permitiendo que las ciudades puedan alcanzar un desarrollo económico sostenible y una mejor calidad de vida.

Este tipo de entorno (ciudades inteligentes) obliga a realizar análisis de datos para tomar decisiones en tiempo real. Las herramientas del ecosistema de Hadoop permiten que sea posible recopilar, almacenar y procesar dichos datos, y aprovecharlos para su **análisis en tiempo real**. Todo esto ha potenciado que cada vez más empresas busquen incluir datos no estructurados, pero potencialmente muy valiosos, junto a sus datos empresariales tradicionales en sus análisis de inteligencia empresarial.

A partir de las muchas definiciones disponibles de *big data*, podemos resumir que se refiere a grandes conjuntos de datos diversos que crecen a un ritmo cada vez mayor. Este concepto abarca el volumen de información, la velocidad a la que se crea y recopila, y la variedad de los datos que se incorporan.

El *big data* es un conjunto de datos demasiado grande, complejos y dinámicos, de forma que no es factible administrarlos y procesarlos de forma eficiente y escalable para cualquier hardware/software convencional.

4.1. Las 5 «V» del *big data*

Para definir las características del *big data*, inicialmente los investigadores plantearon las 3 «V», pero hoy se considera más adecuado plantear las 5 «V»: volumen, velocidad, variedad, veracidad y valor (*volume*, *velocity*, *variety*, *veracity* y *value*).

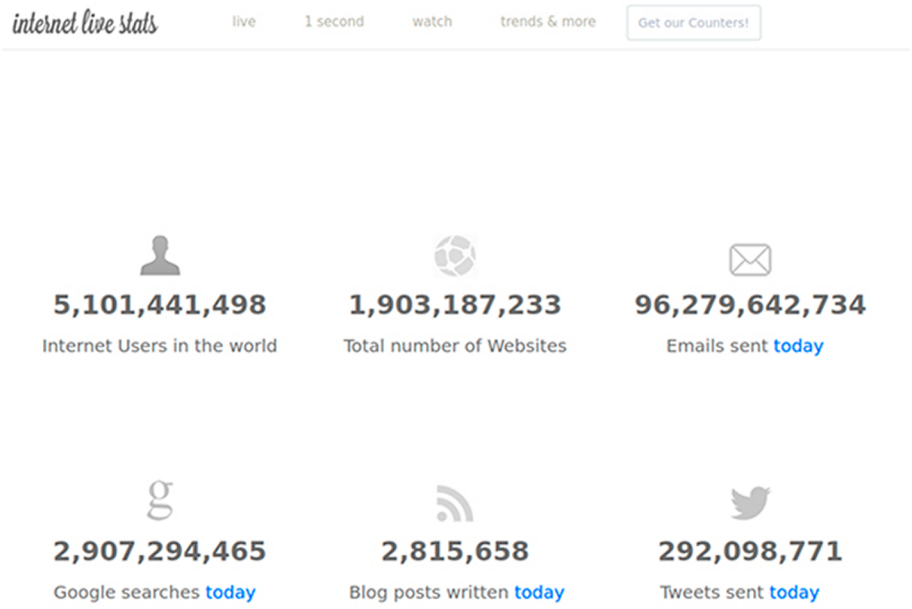
1) Volumen: se refiere a la gran cantidad de datos generados en forma de correos electrónicos, tuits, fotos, videoclips, datos de sensores, etc. Habitualmente se habla de exabytes diarios, zettabytes anuales y pronto de brontobytes. Resulta interesante ver la cantidad de información que se genera en línea en: https://www.internetlivestats.com (figura 6).

Lecturas complementarias

O'Reilly Media (2012, 19 de enero). «Volume, Velocity, Variety: What You Need to Know About Big Data» [en línea]. *Forbes*. <https://www.forbes.com/sites/oreillymedia/2012/01/19/volume-velocity-variety-what-you-need-to-know-about-big-data>

Anil Jain (2016, 17 de septiembre). «The 5 V's of big data» [en línea]. *IBM. Watson Health Perspectives*. <https://www.ibm.com/blogs/watson-health/the-5-vs-of-big-data>

Figura 6. Cantidad de información que se genera en línea



Fuente: <https://www.internetlivestats.com>

Figura 6

La figura muestra el estado a las 08.20 horas del 8 de noviembre de 2021 de los datos generados durante ese día, pero es posible ver las cantidades por segundo o por año.

2) Variedad: se refiere a los diferentes tipos de datos que se generan cada día. En el pasado, el análisis se centraba en datos estructurados (básicamente, en tablas de bases de datos relacionales), pero en la actualidad el 80 % de los datos mundiales no están estructurados y, por tanto, no se pueden procesar fácilmente con las tecnologías tradicionales de bases de datos.

3) Velocidad: se refiere a la velocidad a la que se generan/procesan y analizan los nuevos datos. Resulta interesante observar el tema de la velocidad en contadores como Internetlivestats. Por ejemplo, se puede considerar la velocidad a la que se verifican las transacciones con tarjeta de crédito (en 2014, Visa procesaba 150 millones de transacciones por día, 47.000 por segundo) para detectar actividades fraudulentas o los milisegundos que tardan los sistemas de negociación en analizar y captar movimientos bursátiles que desencadenan decisiones para la compraventa de acciones. Es evidente que no se puede almacenar esta enorme cantidad de datos en bases de datos, por lo que se necesita analizarlos mientras se generan y luego desecharlos.

Lecturas complementarias

El Economista (2014, 28 de mayo). «Data Center de Visa puede procesar 47,000 transacciones por segundo» [en línea]. *El Economista*. México. <https://www.eleconomista.com.mx/tecnologia/Data-Center-de-Visa-puede-procesar-47000-transacciones-por-segundo-20140528-0067.html>

4) Veracidad: se refiere a la incertidumbre o confiabilidad de los datos. Para algunos datos, y sobre todo cuando son en cantidades significativas, la calidad y la precisión son menos controlables.

Publicaciones de Twitter

Consideremos las publicaciones de Twitter: dentro del tuit existen etiquetas, abreviaturas, errores tipográficos y lenguaje coloquial, elementos que pueden poner en entredicho la confiabilidad o precisión del contenido, pues están sujetos a interpretación. Los datos no tienen valor si no son precisos (sobre todo cuando los analizan máquinas), y se debe tener mucho cuidado cuando son estos datos los que forman el conjunto de entrada a sistemas de toma automatizada de decisiones.

5) Valor: el *big data* es en sí mismo interesante, pero a menos que se pueda utilizar para tomar decisiones analíticas con la intención de mejorar algún aspecto del sistema estudiado, no tendrá demasiada utilidad. Si bien es cierto que los datos encierran una buena cantidad de información, el desafío es identificar qué tiene valor y luego transformar y extraer los datos representativos para su análisis. Con ello, las empresas e instituciones pueden desarrollar una comprensión más profunda de su negocio, sus procesos y proyectos, etc., lo que conducirá a un mayor conocimiento, productividad y eficiencia, para que las empresas generen, por ejemplo, una posición competitiva más sólida. Muchos expertos en *big data* consideran que el «valor» es la característica más importante de todas, la que determina el éxito de cualquier proyecto que contemple el *big data* como herramienta.

4.2. Desafíos del *big data*

El crecimiento de los datos en diferentes dimensiones presenta muchos desafíos y oportunidades para las organizaciones. Con el **aumento del volumen y de la velocidad** de generación de los datos, es esencial que se pueda extraer información en tiempo real útil para la toma de decisiones, de lo contrario, los negocio o proyectos correrán el riesgo de verse inundados por una avalancha de datos (*data deluge*) en entornos hipercompetitivos.

Enlace complementario

Podéis consultar la definición de *avalancha de datos* de la Wikipedia en el enlace siguiente: <https://en.wikipedia.org/wiki/Information_explosion>

Por eso el desafío es «atravesar» los enormes volúmenes de datos y acceder al **nivel de detalle** necesario a alta velocidad; a medida que aumente su granularidad, esto será cada vez más complejo. Este tipo de situación deberá ser acompañada por un cambio en la estrategia de negocio, ya que utilizar datos en tiempo real implicará tomar decisiones en tiempo real, pero también para un negocio que se desarrolle en tiempo real.

A esto hay que añadir otros elementos que aumentan la complejidad: además del aumento de la velocidad y de la variedad de los datos, sus **flujos** pueden ser muy inconsistentes, con picos periódicos (diarios, estacionales y desencadenados por eventos como los datos de las redes sociales) que serán muy difíciles de gestionar. Además, los datos son no estructurados (correos electrónicos, fotos, vídeos, dispositivos de monitorización, sensores, PDF, audio, etc.), con lo que la gestión se vuelve extremadamente compleja.

Las variedades de **datos no estructurados** crean problemas de almacenamiento, de minería y de análisis. En dominios como consumo, finanzas, seguridad, medicina, etc., la cantidad de posibles fuentes es extremadamente grande, lo que hace inviable el procesamiento en una sola máquina. Esto obliga a buscar sistemas de administración de datos que puedan manejar bases de datos distribuidas, sin esquemas y no relacionales.

Por otro lado, la **visualización** de datos puede comunicar tendencias y valores atípicos mucho más rápido que las tablas que contienen números y texto, pero se vuelve una tarea difícil cuando se trata de cantidades extremadamente grandes de información o cuando esta proviene de una gran variedad de categorías. Generalmente, los valores atípicos representan alrededor del 1-5 % de los datos, pero cuando se trabaja con cantidades masivas de datos, ver este 1-5 % de los datos puede ser bastante difícil y no es una tarea trivial representar esos puntos sin problemas de visualización.

Otro desafío importante está en la **selección** de los datos. Normalmente se trata de un flujo continuo de datos entrantes, gran parte de los cuales no tienen interés, de manera que pueden filtrarse y comprimirse, o simplemente descartarse. El reto es definir los filtros de forma que no descarten información útil. Por ejemplo, se puede considerar la lectura de un sensor que difiere sustancialmente del resto: puede ser que el sensor esté defectuoso o puede que sea la lectura la que necesita atención. El reto será plantearse cómo deben tomarse

estas decisiones y recurrir, por ejemplo, a técnicas de correlación, ya que generalmente los datos de sensores estarán correlacionados espacial y temporalmente. Es necesario contemplar **técnicas para la reducción del conjunto de datos** para que estos puedan ser procesados en línea, ya que no habrá tiempo para almacenarlos primero y reducirlos después.

A continuación, se verá cómo podemos superar los desafíos que surgen debido al volumen, la variedad y la velocidad a través de ecosistemas como Hadoop, utilizando las bases de datos NoSQL para resolver los problemas que surgen debido a la variedad.

4.3. Arquitectura para *big data*

Como en cualquier entorno orientado a datos, una arquitectura para el procesamiento de *big data* está formada por cuatro niveles que interactúan, más uno global de administración:

1) Recolección de datos.
2) Almacenamiento.
3) Procesamiento.
4) Visualización.

Esta arquitectura es la habitual en cualquier entorno orientado a datos (por ejemplo, minería de datos, inteligencia empresarial o aprendizaje profundo). No obstante, el concepto de *big data* ha hecho que evolucione cada uno de estos niveles, generando nuevas herramientas, algoritmos y entornos para adecuarse a los datos masivos.

1) En la **recolección de los datos** orientados al *big data*, han surgido una gran cantidad de herramientas orientadas a este fin (*data ingestion*), que permiten el procesamiento secuencial habitual de datos almacenados (lotes o *batch*) para obtener el siguiente tramo o secuencia (*chunk*) de datos desde el último leído; en todo caso, en su mayoría, y dado el volumen de los mismos, las herramientas han evolucionado para recolectar eficientemente flujos de datos (*streams*) en tiempo real.

2) En cuanto al **almacenamiento**, también han surgido grandes cambios para adaptarse a las características de los datos; ello se ha reflejado en una gran cantidad de desarrollos de motores de bases de datos (NoSQL, documentos, columnas, llave/valor, grafos) y sistemas de archivos distribuidos para adecuarse a la realidad de procesamiento y, así, disponer de escalabilidad, fiabilidad y cercanía de los datos para su tratamiento (por ejemplo, Apache HDFS, BeeGFS, DiscoDDFS, Google GFS, BaiduFS, GlusterFS, QuantcastFS, CephFS, GridGain-in-memoryFS, LustreFS).

3) Para el **procesamiento**, como ya hemos comentado, se han diseñado nuevos paradigmas que han caracterizado todo el entorno (y cada una de las capas para adecuarlas a sus necesidades); se conocen como MapReduce (MR) o *massive parallel processing* (MPP).

a) MapReduce es un paradigma de programación cuyo nombre viene dado por las dos funciones que lo componen (*map* y *reduce*), y se implementa en el entorno Apache Hadoop (código abierto). Este paradigma, si bien no es la solución para todos los problemas, trabaja con grandes conjuntos de datos (petabytes) y se ejecuta sobre sistemas de archivos distribuidos (HDFS) y vinculados a herramientas como HBase, Hive, Impala o Cassandra (bases de datos).

Se trata de un paradigma apto para procesar los datos que pueden trabajar sobre tuplas del tipo [clave, valor] y donde la función `map()` procesa en paralelo las tuplas de un dominio y genera una lista de pares en un dominio diferente, las cuales se agruparán bajo la misma clave utilizando un esquema de procesamiento *master-worker* en forma de árbol.

Posteriormente, la función `reduce()` se aplica de forma paralela para cada grupo y genera una colección de valores para cada dominio. Un ejemplo típico es el procedimiento para contar el número de veces que aparecen las palabras en un texto. Las funciones serían:

```
map(string name, string document):
    foreach word w in document:
        generate-tupla(w, 1);

reduce(string word, iterator partialList):
    int total = 0;
    foreach value in partialList:
        total += int(value);
    generate(word, total);
```

Función `map()`

La función `map()` divide el documento por palabras y genera las tuplas [palabra, valor]. Por ejemplo, la famosa frase de Alan Turing «La ciencia es una ecuación diferencial. La religión es una condición de frontera» quedará: [La,1], [ciencia,1], [es,1], [una,1], [ecuación,1], [diferencial,1], [La,1], [religión,1], [es,1], [una,1], [condición,1], [de,1], [frontera,1]. El entorno agrupará todas las claves iguales ['La,1,1', 'es,1,1', 'una,1,1' y el resto 'clave,1'] como entrada a reduce(), que generará la agrupación y la suma de los valores de las claves, que en nuestro ejemplo será [La,2], [es,2], [una,2] y el resto [clave,1].

b) MPP ofrece una solución tradicional adaptada al *big data* basada en dividir los grandes conjuntos de datos en secciones (*slices*) que serán más fáciles de gestionar; cada uno de ellos será asignado a un elemento de cómputo para su procesamiento. El dispositivo de cómputo los procesará y, cuando termine, el sistema combinará los resultados parciales para dar un resultado final (equivalente a una secuencia *fork-join* en un modelo *master-workers*). Dado que los elementos de cómputo (procesadores) trabajarán sobre su segmento de datos asignado de la BD y se comunicarán a través de mensajes cuando generen los resultados, no hay interacción entre ellos; esto se conoce como «débilmente acoplados» (otros autores lo denominan *shared nothing*).

Enlace complementario

Podéis consultar la definición del modelo *fork-join* de la Wikipedia en el enlace siguiente:

<https://en.wikipedia.org/wiki/Fork–join_model>

Entre las características principales de estos sistemas, se pueden citar sus **posibilidades de sintonización y escalado ilimitado** (en principio, al número de nodos disponibles), con el consiguiente incremento de su rendimiento (prácticamente de forma lineal) y sin cuellos de botella.

Existe gran cantidad de bibliografía sobre la comparación entre MR y MPP. Sin embargo, muchas veces la solución no proviene de uno u otro paradigma, y dado que se complementan bien en cuanto a las prestaciones y a los tipos de datos que obtienen mejores rendimientos, es habitual que se utilicen arquitecturas mixtas en las que generalmente primero se aplica MR sobre datos no estructurados y, posteriormente, MPP para procesar y visualizar los datos estructurados generados en la primera operación.

Lectura complementaria

Un ejemplo de bibliografía sobre la comparación entre MR y MPP es el siguiente:

Alexey Grishchenko (2015, 13 de julio). «Hadoop vs MPP» [en línea]. *Distributed Systems Architecture*. <https://0x0fff.com/hadoop-vs-mpp>

4) Finalmente, en el último nivel de la arquitectura, se encuentra la **visualización** de *big data*. Se refiere a la implementación de técnicas de visualización usando estructuras y gráficos adecuados para obtener «información» a diferentes niveles sobre las relaciones existentes en los datos. Las estrategias de visualización son diversas e incluyen aplicaciones que pueden mostrar cambios en tiempo real y gráficos más ilustrativos, dejando de lado los típicos gráficos de dos variables (circulares, de barras o de líneas). Estas visualizaciones derivan en una representación «más atractiva, más visual» de los datos, considerada por algunos autores como «una visualización más artística y menos artesanal» de los datos.

Normalmente, cuando las empresas o instituciones necesitan presentar relaciones entre datos, utilizan gráficos de barras y diagramas con una variedad de colores, términos y símbolos. Sin embargo, el principal problema con este tipo de visualización es que generalmente no funciona cuando se quieren presentar datos muy grandes o muy variados (desde valores muy pequeños a muy grandes), o datos que no tienen la misma unidad o con cuatro parámetros, por ejemplo.

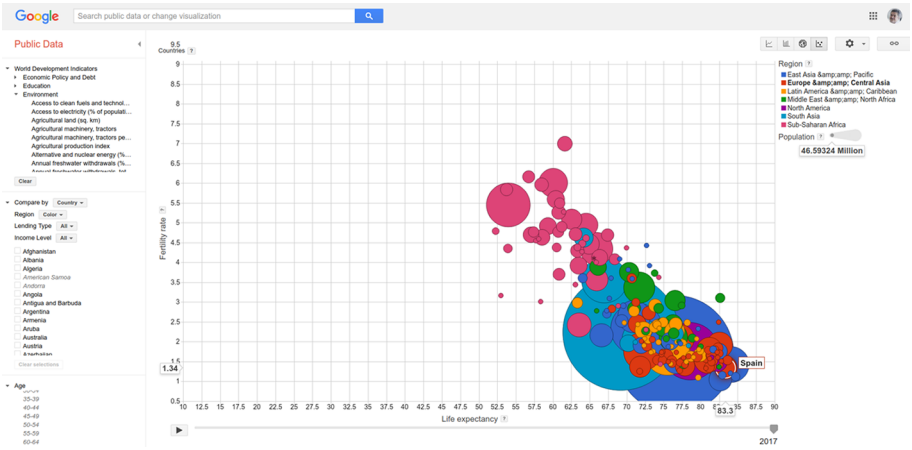
La visualización de *big data* está orientada a utilizar ilustraciones gráficas más interactivas, que cuenten con personalización y animación, para mostrar figuras y establecer conexiones entre piezas de información, y que sea el usuario quien escoja la escala o el grado de detalle que quiere ver.

Como ejemplo, se puede mencionar (y se recomienda interactuar con este entorno de visualización de datos abiertos) el **Google Public Data Explorer**. En el ejemplo de la figura 7, se muestra la relación de la esperanza de vida y la fertilidad por países, incluyendo su población, y es el usuario quien decidirá qué desea ver: la evolución desde 1960 a 2017 (o seleccionar cualquier otra variable para insertar en el gráfico).

Enlace complementario

Google Public Data Explorer: <https://www.google.com/publicdata/ directory>

Figura 7. Ejemplo de visualización de *big data* con Google Public Data Explorer



Herramientas para visualizar *big data*

Existe una gran cantidad de herramientas para visualizar el *big data*: Chartsblocks (http://www.chartblocks.com), Databox (https://www.adamenfroy.com/recommends/data-box), DataHero (https://datahero.com), Datawrapper (https://datawrapper.de), Plotly (https://plot.ly/), PowerBI (https://powerbi.microsoft.com) y Qlik (https://www.qlik.com); y para desarrolladores (aquellas utilizadas por diseñadores web o aplicaciones para visualizar *big data*): Chart.js (http://www.chartjs.org), Chartist.js (https://gionkunz.github.io/chartist-js), D3.js (http://d3js.org), Ember Charts (http://addepar.github.io/ember-charts), FusionCharts (https://www.fusioncharts.com), Google Charts (https://developers.google.com/chart), Highcharts (https://www.highcharts.com), n3-charts (https://github.com/n3-charts), NVD3.js (http://nvd3.org) y Processing.js (https://processing.org), entre otras.

4.4. Herramientas

El proyecto Apache™ Hadoop® es una plataforma de código abierto para el cómputo distribuido, confiable y escalable, utilizada por una gran cantidad de empresas e instituciones, que implementa el algoritmo de MapReduce.

Apache Hadoop es un entorno para el procesamiento distribuido de grandes conjuntos de datos a través de clústeres de cómputo. Utiliza modelos de programación sencillos y tiene la posibilidad de escalar desde servidores individuales a miles de procesadores.

Básicamente, Hadoop (V2.x o V3.x) está formado por cuatro módulos:

1) HadoopYARN: entorno para la planificación de tareas y la gestión de recursos del clúster.

2) Hadoop MapReduce: sistema basado en YARN para el procesamiento paralelo de grandes conjuntos de datos.

3) Hadoop Distributed FileSystem (HDFS): sistema de archivos distribuido que proporciona acceso de alto rendimiento a los datos de la aplicación.

4) Hadoop Common: las utilidades comunes que soportan los otros módulos de Hadoop.

No obstante, Hadoop se puede relacionar, integrar o ampliar de forma fácil y rápida con los siguientes proyectos (solo algunos de los más habituales o referenciados):

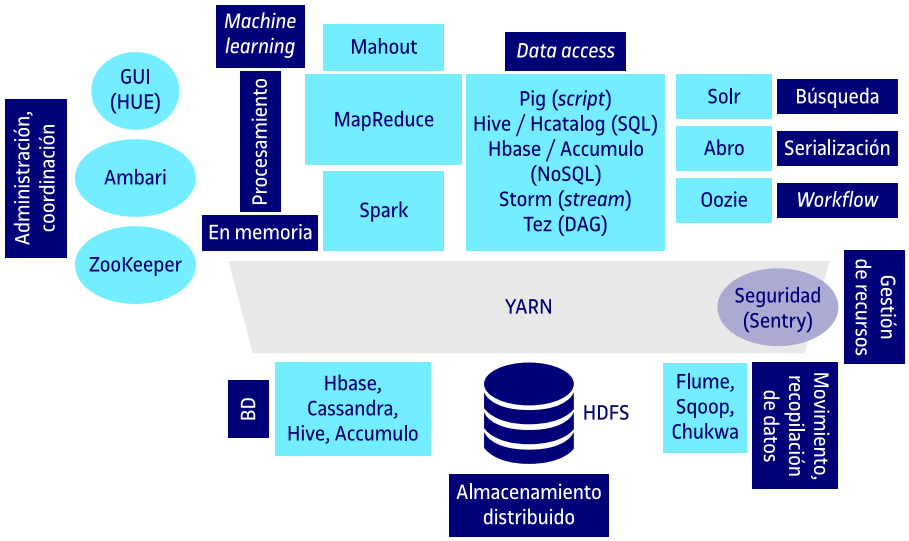
- **Ambari**: herramienta web para el aprovisionamiento, la gestión y la monitorización de clústeres Hadoop con HDFS, MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig y Sqoop.
- **Avro**: serialización de datos.
- **Cassandra**: base de datos escalable sin puntos de fallo únicos.
- **Chukwa**: recopilación de datos en sistemas distribuidos.
- **Drill:** *SQL query engine* de baja latencia que soporta aplicaciones distribuidas para el análisis interactivo de *datasets*.
- **Flume**: sistema distribuido para recoger, agregar y mover grandes cantidades de datos, desde diferentes fuentes y también de *datastores* centralizados.
- **HBase**: base de datos orientada a columnas distribuida y escalable.
- **Hive**: almacén de datos que proporciona resumen de datos y consultas *ad hoc*.
- **Mahout**: entorno de aprendizaje automático y minería de datos.
- **Pig**: lenguaje de flujo de datos de alto nivel.
- **Spark**: motor de cálculo en memoria especializado en procesar datos de flujos de datos (*streams*).
- **Sqoop**: herramienta para transferir datos entre Hadoop y bases de datos estructuradas, como, por ejemplo, bases de datos relacionales.

- **Tez**: entorno de programación de flujo de datos generalizado para grafos dirigidos.

- **ZooKeeper**: servicio de coordinación de alto rendimiento para aplicaciones distribuidas.

La figura 8 muestra un entorno Hadoop con sus actores más importantes y su rol (uno de los posibles) dentro de la plataforma.

Figura 8. Entorno Hadoop



4.5. NoSQL

Un término importante en el contexto del *big data* es ***NoSQL* (no solo SQL)**, que abarca una amplia variedad de tecnologías de bases de datos diferentes que son **no relacionales, sin esquema, distribuidas y escalables horizontalmente**. Están diseñadas para hacer frente al *big data* y a las aplicaciones web en tiempo real que requieren análisis de tipos de datos dispares y de un volumen extremadamente alto.

Muchos de los sistemas NoSQL no proporcionan garantías de atomicidad, consistencia, aislamiento y durabilidad, a diferencia de los sistemas de bases de datos relacionales, pero esto no significa un problema en el procesamiento del *big data*. Las bases de datos relacionales, por otro lado, no están diseñadas para hacer frente a los desafíos de escala y agilidad de las aplicaciones actuales, ni para aprovechar el almacenamiento de bajo coste ni la potencia de procesamiento distribuida disponible en la actualidad.

Atomicidad

Característica que asegura que una operación en la BD se ha realizado o no, y, por lo tanto, ante un fallo del sistema, no puede quedar a medias.

Hoy en día existen más de 250 bases de datos NoSQL (http://nosql-database.org), con diferentes visiones y metodologías para el almacenamiento de datos diferentes. Existen varios enfoques para clasificar las bases de datos NoSQL, cada uno con diferentes categorías (y subcategorías), pero la clasificación más utilizada es la que se basa en el **modelo de datos**. En la tabla 1 mostramos algunos ejemplos de BD de estos modelos (aunque algunos de ellos pueden situarse en más de una categoría).

Lectura complementaria

Susan George (2013, julio). «NOSQL - NOTONLY SQL» [en línea]. *International Journal of Enterprise Computing and Business Systems* (vol. 2, núm. 2). <http://www.ijecbs.com/July2013/3.pdf>

Tabla 1. Ejemplos de bases de datos de los diferentes modelos de datos

Modelo	Bases de datos
Columna	Hbase, Cassandra, Accumulo, Hypertable2.
Documento	MongoDB, CouchDB, RavenDB, RethinkDB, Terrastore 3.
Grafo	Neo4J, AllegroGraph, Infinite Graph, HyperGraphDB 5.
Valor clave	Dynamo, Riak, Redis, MemcacheDB, Voldemort, Scalaris, BerkeleyDB, SimpleDB 4.
Multimodelo	CortexDB, AlchemyDB.

Estos modelos de datos tienen las características siguientes:

1) Columna: los datos se almacenan en celdas agrupadas en columnas de datos, en lugar de en filas. Las columnas se agrupan lógicamente en familias de columnas –que pueden contener una cantidad prácticamente ilimitada de columnas–, que pueden ser creadas en tiempo de ejecución o en la definición del esquema.

La lectura y escritura se realiza utilizando columnas en lugar de filas, y en comparación a la mayoría de los sistemas de bases de datos relacionales, almacenan datos en columnas. El beneficio de almacenar datos en columnas es que la búsqueda y el acceso son rápidos, y la agregación de datos sumamente eficiente.

Las bases de datos relacionales almacenan una sola fila como una entrada de disco continua y las diferentes filas se almacenan en diferentes lugares del disco, mientras que las bases de datos en columnas almacenan todas las celdas correspondientes a una columna como una entrada de disco continua, lo que agiliza la búsqueda/acceso.

Cassandra

Cassandra es capaz de manejar aplicaciones comerciales que requieren escalabilidad masiva, disponibilidad continua, alto rendimiento, seguridad y simplicidad operativa. Empresas como Netflix, Sky, SoundCloud, Healthx, GoDaddy o eBay, entre otras, utilizan Cassandra.

2) Documento: el concepto central de un almacén de documentos es la noción de *documento*, cuyos datos son una colección de pares clave-valor. Las codificaciones estándar comunes son realizadas en formatos XML, YAML y JSON (o también en formas binarias como BSON).

Una diferencia importante entre un almacén de valores clave y un almacén de documentos es que este último incorpora metadatos de **atributos asociados** con el contenido almacenado, lo que esencialmente proporciona una forma de consultar los datos en función del contenido.

MongoDB

MongoDB se utiliza para inteligencia operativa y análisis en tiempo real. Diversas empresas han construido su *suite* de Internet de las cosas (IoT) en MongoDB, lo que ha llevado el poder del *big data* a una nueva gama de aplicaciones de Internet industrial, que incluyen fabricación, automoción, comercio minorista o energía, entre otras.

3) Grafo: está diseñado para datos cuyas relaciones están bien representadas como un grafo, es decir, elementos interconectados con un número indeterminado de relaciones entre ellos. Una base de datos de grafos es esencialmente una colección de nodos y conexiones.

Cada **nodo** representa una entidad y cada **conexión** representa una relación entre dos nodos. Cada nodo se define mediante un identificador único, un conjunto de conexiones salientes o entrantes y un conjunto de propiedades expresadas como pares clave-valor. Cada conexión está definida por un identificador único, un nodo de lugar de inicio o lugar de finalización y un conjunto de propiedades.

Este tipo de datos pueden ser relaciones sociales, enlaces de transporte público, mapas de carreteras o topología de redes, o cualquier otra relación entre el nodo de entrada y el de salida. Las bases de datos basadas en grafos son adecuadas para extraer datos de las redes sociales y muy útiles para trabajar con datos en disciplinas que implican relaciones complejas y esquemas dinámicos, como la gestión de la cadena de suministro, los sistemas biológicos y los sistemas de recomendación.

4) Valor clave: es uno de los modelos de datos no triviales más simples en formato y sin esquema. La clave puede ser sintética o autogenerada, mientras que el valor puede ser una cadena de caracteres (*string*), objetos JSON o BLOB (objeto grande básico), etc. Generalmente, utiliza una tabla ***hash*** en la que existe una clave única y un puntero a un elemento de datos en particular.

Hash

Función también llamada *resumen*, que convierte una entrada de letras y números de cualquier tamaño en una salida única de longitud fija.

Riak

Riak se puede utilizar en aplicaciones que necesitan clave de sesión en tiendas de comercio electrónico; es utilizada por empresas como Best Buy, Copious, Ideel, Shopzilla.

5) Multimodelo: las bases de datos multimodelo soportan combinaciones de los modelos antes mencionados o incorporan también el modelo clásico SQL. Por ejemplo, MariaDB es en su diseño una BD SQL, pero soporta tablas NoSQL y almacenamiento por columnas, siendo considerada por muchos expertos como una BD multimodelo.

4.6. Nube pública y herramientas para *big data*

Hoy en día, la gran mayoría de proveedores de nube ofrecen entornos preparados para procesar *big data*. Citaremos algunos ejemplos (la lista no es exhaustiva):

1) Amazon EMR (https://aws.amazon.com/emr): plataforma de *big data* en la nube que permite procesar grandes cantidades de datos utilizando herramientas de código abierto, como Apache Spark/Hadoop, Apache Hive, Apache HBase, Apache Flink, Apache Hudi y Presto. Con el entorno EMR se pueden realizar análisis a escala de petabytes con un coste un 50 % menor con respecto a las soluciones locales tradicionales (según el proveedor). Para trabajos de ejecución corta, se puedan activar y desactivar clústeres, y pagar solo por los segundos que se han usado las instancias. Para cargas de trabajo de larga duración, se pueden crear clústeres de alta disponibilidad que escalen automáticamente para satisfacer la demanda.

2) Azure HDInsight (https://azure.microsoft.com/en-us/services/hdinsight): permite ejecutar tareas en entornos de código abierto, como Apache Hadoop, Spark, Kafka, Hive, Hbase, Stom y ML. Acepta tareas de diferente tipo, tanto empresariales como de investigación, de forma que mantiene la rentabilidad y procesa sin esfuerzo grandes cantidades de datos mediante el uso de las plataformas de código abierto ya mencionadas. Con HDInsight se pueden generar rápidamente clústeres de *big data* bajo demanda, escalar (o reducir) según las necesidades de uso y pagar por solo aquello que se ha utilizado. La plataforma garantiza la privacidad de los datos mediante una red virtual de Azure, y se integra con Azure Active Directory, Data Factory y Data Lake Storage, lo que permite crear canales de análisis completos encriptados.

3) Google Dataproc: es una infraestructura integrada dentro de Google Cloud que permite el procesamiento de *big data* y análisis a través de herramientas de código abierto de forma rápida, fácil y segura en la nube. Al igual que la mayoría de proveedores de nube, tiene cuentas de uso gratuito, con un determinado crédito y tiempo limitado (en Google son 300 dólares y un periodo de prueba de 90 días), y también permite el uso gratuito (hasta límites mensuales) de algunos productos como BigQuery. Dataproc permite crear clústeres para ejecutar Apache Spark/Hadoop o Presto, entre otros, pagando solo por uso (segundos), con cifrados para mantener la confidencialidad de los datos y con una seguridad unificada integrada en cada clúster, con escalado automático y eliminación de clústeres inactivos.

Enlace complementario

Google Dataproc: <https://cloud.google.com/dataproc/>

BigQuery

Almacén de datos sin servidor totalmente administrado que permite un análisis de datos escalable en petabytes y funciona como plataforma como servicio que admite consultas mediante ANSI SQL.

4) Alibaba Cloud Elastic MapReduce (EMR) (https://www.alibabacloud.com/products/emapreduce): es una solución de procesamiento de *big data* que se ejecuta en la plataforma Alibaba Cloud. EMR se basa en instancias ECS de Alibaba Cloud y en Apache Hadoop y Apache Spark, lo que permite la utilización de los componentes del ecosistema de Hadoop y Spark, como Apache Hive, Apache Kafka, Flink, Druid y TensorFlow, entre otros. Con EMR se pueden analizar y procesar datos almacenados en diferentes servicios de almacenamiento de datos en la nube de Alibaba, como Object Storage Service (OSS), Log Service (SLS) y Relational Database Service (RDS).

5) Cloudera (https://www.cloudera.com/downloads.html): proporciona una plataforma de software para ingeniería de datos, almacenamiento de datos, aprendizaje automático y análisis, que se ejecuta en la nube o en las instalaciones propias de la empresa o institución. Cloudera comenzó como una distribución híbrida de código abierto de Apache Hadoop (CDH), que facilitaba el uso de Hadoop de clase empresarial, aunque luego ha ido incluyendo diversos proyectos de código abierto con licencia de Apache (Apache Spark, Hive, Avro, HBase, etc.) que se combinan para formar la plataforma Apache Hadoop. Cloudera, que también es patrocinador de la Apache Software Foundation, ha realizado alianzas con otras empresas (Azure, EMC, etc.) o directamente las ha comprado, como HortonWorks (2019) o Arcadia Data, para transformarse en una de las empresas líderes del mercado en soluciones *in-situ* o en la nube de plataformas de *big data*.

6) MapR (hoy, HPE Ezmeral Data Fabric) (https://www.hpe.com/us/en/software/data-fabric.html): MapR fue otro de los grandes pilares para proveer soluciones empresariales para el procesamiento de *big data* a partir de software de código abierto basadas en Apache Hadoop y Spark, un sistema de archivos distribuido, un sistema de administración de base de datos de múltiples modelos y el procesamiento de flujo de eventos para el análisis de datos en tiempo real. La plataforma MapR se ejecuta tanto en hardware propietario de las empresas o instituciones como en servicios de cómputo en la nube pública. En agosto de 2019, tras las dificultades financieras, la tecnología y la propiedad intelectual de la empresa fueron adquiridas por Hewlett Packard Enterprise para conformar la plataforma Ezmeral Data Fabric de HP.

Dado que se trata de un sector muy dinámico, surgen y desaparecen empresas, o son compradas por otras; algunas de las que ofrecen soluciones son, entre otras, las siguientes:

- Domino (https://www.dominodatalab.com/product/domino-data-science-platform/)
- Datameer (https://www.datameer.com)

- GreenPlum, comprada por VMWare recientemente y rebautizada como Tanzu (https://greenplum.org)
- SAP (https://www.sap.com/index.html)
- IBM (https://www.ibm.com/analytics/hadoop)
- QuBole (https://www.qubole.com/platform/open-data-lake-platform)
- HPCC (https://hpccsystems.com/try-now)
- RapidMiner (https://rapidminer.com/get-started/)
- Teradata (https://www.teradata.com)
- Tableu (https://www.tableau.com/why-tableau)

No obstante, el estudiantado o la persona usuaria siempre puede ir a la fuente e instalar el ecosistema Apache Hadoop desde los repositorios oficiales o utilizar una plataforma creada para probar toda esta tecnología, llamada Bigtop, que incluye (bigtop 1.4.0 stack): alluxio, ambari, apex, bigtop_groovy, bigtop_jsvc, bigtop_tomcat, bigtop_utils, flink, flume, giraph, gpdb, hadoop, hama, hbase, hive, ignite_hadoop, kafka, mahout, oozie, phoenix, qfs, solr, spark, sqoop, sqoop2, tajo, tez, ycsb, zeppelin y zookeeper.

Enlaces complementarios

Repositorios oficiales: <https://projects.apache.org/projects.html>

Bigtop: <https://cwiki.apache.org/confluence/ pages/viewpage.action?pageId=70256303>

5. Modelos de interacción (API)

Las interfaces de programación de aplicaciones (API, *application programming interface*) son un modelo de interacción e intercambio de datos entre aplicaciones que simplifican el desarrollo y la innovación de software, ya que permiten que las propias aplicaciones intercambien datos y funcionalidades de manera fácil y segura (IBM).

Enlace complementario

En la web de IBM encontraréis información sobre qué es una API. Podéis consultar el enlace siguiente:

<https://www.ibm.com/cloud/learn/api>

Esto permite a las empresas abrir los datos y la funcionalidad de sus aplicaciones a desarrolladores externos, socios comerciales o departamentos internos dentro de sus empresas de forma estandarizada y segura, lo cual posibilita que los diferentes servicios se comuniquen entre sí y se reutilicen los datos de forma dinámica mediante una interfaz documentada. Los desarrolladores simplemente consultan la documentación de la API y ya pueden interactuar con servicios, un recurso que es cada vez más utilizado (por ejemplo, muchas de las aplicaciones web más populares de la actualidad no serían posibles sin las API).

Por su estructura interna, una API no es más que un conjunto de reglas definidas que explican cómo las aplicaciones se comunican entre sí en un modelo de interacción cliente-servidor.

Funcionamiento de una API

Una aplicación web cliente inicia una llamada a la API para recuperar información del servidor, indica en la URL los parámetros de la información que desea obtener y, después de verificar la validez del servidor, obtiene la información solicitada y la devuelve a la aplicación cliente.

Las llamadas a la API generalmente incluyen **credenciales de autorización** para reducir el riesgo de ataques al servidor. El acceso se puede limitar para minimizar las amenazas a la seguridad; además, durante el intercambio, los encabezados HTTP, las *cookies* o los parámetros de la cadena de consulta proporcionan capas de seguridad adicionales a los datos.

Existen diversos **protocolos** para proporcionar a los usuarios un conjunto de reglas definidas que especifican los tipos de datos y comandos aceptados, entre los que destacan los siguientes:

- **SOAP (*simple object access protocol*)**: es un protocolo para generar APIs, que utiliza XML como formato de datos y HTTP/HTTPS como protocolo de transmisión. Su diseño data de 1998 y es un protocolo flexible, puesto que puede utilizar otros protocolos para la transmisión de información si es necesario. Si bien su uso es frecuente, no es el más utilizado, ya que su dependencia de XML lo hace muy estricto, debido a que depurar código XML

resulta complejo. Muchos programadores prefieren XML-RPC, anterior a SOAP, o su alternativa JSON-RPC, Tanto uno como otro utilizan el mecanismos de RPC (<https://en.wikipedia.org/wiki/Remote_procedure_call>), pero el primero utiliza XML mientras que el segundo JSON para codificar los datos. Estos protocolos son mucho más simples y ligeros, motivo por el cual utilizan menos recursos, pero no tienen la versatilidad de SOAP.

- **REST (*representational state transfer*)**: este protocolo elimina la dependencia de un lenguaje, como XML, y permite la codificación de los datos en múltiples formatos, siendo JSON el más utilizado; utiliza HTTP/HTTPS como protocolo de transmisión. Las APIs que emplean el protocolo REST se denominan *API RESTful* y tienen una arquitectura cliente-servidor sin estado, esto implica que no se almacenan datos de clientes entre las solicitudes GET –que serán diferentes y desconectadas entre sí–, por lo tanto, el programador obtiene más libertad en la implementación, y la interacción es simple y escalable. El protocolo asigna a cada operación una URL única, por lo que el servidor que recibe una solicitud puede determinar qué instrucciones ejecutar para satisfacerla. Si bien las API REST son las más utilizadas hoy en día, para muchas aplicaciones las API JSON-RPC, teniendo en cuenta que tienen un alcance limitado, mejoran el rendimiento de una API REST, por lo tanto, son una alternativa cuando las funcionalidades de esta API pueden satisfacer las necesidades del diseño de la aplicación.

- **gRPC (*Google remote procedural call*)**: API *open-source* desarrollada por Google en 2015 que también utiliza RPC, con la gran ventaja que permite a los desarrolladores definir sus propias funciones para habilitar la comunicación entre servicios según sea necesario. gRPC usa HTTP como capa de transporte incluyendo un conjunto interesante de funciones adicionales (autenticación, control de flujo, etc).

- **GraphQL**: desarrollada por Facebook en 2015, es un lenguaje de consulta y manipulación de datos para APIs, y un entorno de ejecución para realizar consultas de datos. En realidad, es una API para entornos web comparable, en ciertos aspectos, con REST y otras arquitecturas de servicio web. Permite a los clientes definir la estructura de datos requerida, y la misma estructura de datos será retornada por el servidor, aunque esto tienen implicaciones, según muchos expertos, en la eficiencia en la caché web de los resultados de estas consultas. La flexibilidad y variedad del lenguaje de consulta añade complejidad, que puede no resultar atractiva en comparación con otras APIs, como JSON-RPC o la propia REST.

- **Apache Thrift**: desarrollada por Facebook con el objetivo de habilitar la comunicación con servicios escritos en diferentes lenguajes y la escalabilidadnota.[4] Thrift es un lenguaje de definición de interfaz y un protocolo de comunicación utilizado para definir y crear servicios en diversos lenguajes de programación. En esencia es una implementación de un *framework* RPC que utiliza un motor de generación de código combinado con una pila de

(4)Actualmente, es un entorno *open-source* gestionado por la fundación Apache: <https://thrift.apache.org>.

software (*software stack*) para habilitar una API, en la que la pila ayuda a escribir código para definir el lado del cliente y del servidor. La sintaxis del código (en archivos Thrift) es flexible e intuitiva, y será posteriormente la entrada al motor de generación que creará el código requerido en cualquier lenguaje de programación especificado por el desarrollador.

Por ejemplo, para interactuar con la API de GitHub es sumamente fácil a través del comando `curl`:

```
curl https://api.github.com/users/rsuppi
```

Que nos devolverá:

```
{
  "login": "rsuppi",
  "id": ............,
  "node_id": ".................",
  "avatar_url": "https://avatars.githubusercontent.com/u/............?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/rsuppi",
  "html_url": "https://github.com/rsuppi",
  "followers_url": "https://api.github.com/users/rsuppi/followers",
  "following_url": "https://api.github.com/users/rsuppi/following{/other_user}",
  "gists_url": "https://api.github.com/users/rsuppi/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/rsuppi/starred{/owner}{/repo}",
  "subscriptions_url": "https://api.github.com/users/rsuppi/subscriptions",
  "organizations_url": "https://api.github.com/users/rsuppi/orgs",
  "repos_url": "https://api.github.com/users/rsuppi/repos",
  "events_url": "https://api.github.com/users/rsuppi/events{/privacy}",
  "received_events_url": "https://api.github.com/users/rsuppi/received_events",
  "type": "User",
  "site_admin": false,
  "name": null,
  "company": null,
  "blog": "",
  "location": null,
  "email": null,
  "hireable": null,
  "bio": null,
  "twitter_username": null,
  "public_repos": 5,
  "public_gists": 0,
  "followers": 0,
  "following": 0,
  "created_at": "2016-12-15T12:29:58Z",
```

```
    "updated_at": "2021-02-10T08:59:21Z"
}
```

Este es el resultado de toda la información que se puede extraer sin autentificarse del usuario rsuppi.

Enlaces complementarios

API de GitHub: <https://docs.github.com/en/rest/guides/ getting-started-with-the-rest-api>

Introducción a `curl` usando la API de GitHub.: <https://gist.github.com/btoone/2288960>

Bibliografía

Bibliografía básica

Fahad Akhtar, Syed Muhammad (2018). *Big Data Architect's Handbook: A guide to building proficiency in tools and systems used by leading big data experts*. Birmingham: Packt Publishing.

Gupta, Sumit; Saxena, Shilpi (2016). *Real-Time Big Data Analytics*. Birmingham: Packt Publishing.

Salvador, Jaime; Ruiz, Zoila; García-Rodríguez, Jose (2017). «Big Data Infrastructure: A Survey» [en línea]. En: José Manuel Ferrández Vicente; José Ramón Álvarez-Sánchez; Félix de la Paz López; Javier Toledo Moreo; Hojjat Adeli (eds.). *Biomedical Applications Based on Natural and Artificial Computing* (págs. 249-258). International Work-Conference on the Interplay Between Natural and Artificial Computation, La Coruña, 19-23 de junio. <https://link.springer.com/chapter/10.1007%2F978-3-319-59773-7_26> y <http://www.dspace.uce.edu.ec/bitstream/25000/13653/1/Big%20data%20infrastructure%20a%20survey.pdf>

Referencias bibliográficas

Burt, Jeff (2014, 29 de enero). «Cisco Moving Apps to the Network Edge for Internet of Things» [en línea]. *eWeek*. <https://www.eweek.com/networking/cisco-moving-apps-to-the-network-edge-for-internet-of-things>

El Economista (2014, 28 de mayo). «Data Center de Visa puede procesar 47,000 transacciones por segundo» [en línea]. *El Economista*. México. <https://www.eleconomista.com.mx/tecnologia/Data-Center-de-Visa-puede-procesar-47000-transacciones-por-segundo-20140528-0067.html>

Fisher, David Edward; Yang, Shuhui (2016). «Doing More with the Dew: A New Approach to Cloud-Dew Architecture» [en línea]. *Open Journal of Cloud Computing* (vol. 3, núm. 1). <http://www.ronpub.com/publications/OJCC_2016v3i1n02_Fisher.pdf>

Foro Económico Mundial (2012). *Big Data, Big Impact: New Possibilities for International Development* [en línea]. <https://www3.weforum.org/docs/WEF_TC_MFS_BigDataBigImpact_Briefing_2012.pdf>

Foster, Ian; Kesselman, Carl (2014). *The History of the Grid* [en línea]. <http://www.ianfoster.org/wordpress/wp-content/uploads/2014/01/History-of-the-Grid-numbered.pdf>

García Lopez, Pedro y otros (2015, octubre). «Edge-centric Computing: Vision and Challenges» [en línea]. *ACM SIGCOMM Computer Communication Review* (vol. 45, núm. 5, págs. 37-42). <https://doi.org/10.1145/2831347.2831354>

George, Susan (2013, julio). «NOSQL - NOTONLY SQL» [en línea]. *International Journal of Enterprise Computing and Business Systems* (vol. 2, núm. 2). <http://www.ijecbs.com/July2013/3.pdf>

Gilder, George (2013). *Knowledge and Power: The Information Theory of Capitalism and How It is Revolutionizing Our World*. Washington, D. C.: Regnery Gateway.

Grishchenko, Alexey (2015, 13 de julio). «Hadoop vs MPP» [en línea]. *Distributed Systems Architecture*. <https://0x0fff.com/hadoop-vs-mpp>

Gupta, Sushant (2020, 2 de septiembre). «What is big data analytics? Beginner guide to the world of big data» [en línea]. *Tricky Enough*. <https://www.trickyenough.com/big-data-analytics>

Haefele, Paul (2012). «EC2 is 380% more expensive than internal cluster» [en línea]. *Deep Value*. <http://deepvalue.net/ec2-is-380-more-expensive-than-internal-cluster>

Jain, Anil (2016, 17 de septiembre). «The 5 V's of big data» [en línea]. *IBM. Watson Health Perspectives*. <https://www.ibm.com/blogs/watson-health/the-5-vs-of-big-data>

Khan, Atta ur Rehman; Othman, Mazliza; Madani, Sajjad Ahmad; Khan, Samee Ullah (2014). «A Survey of Mobile Cloud Computing Application Models» [en línea]. *IEEE Communications Surveys & Tutorials* (vol. 16, núm. 1). <http://ieeexplore.ieee.org/document/6553297>

Mell, Peter; Grance, Timothy (2011, septiembre). «The NIST Definition of Cloud Computing» [en línea]. *National Institute of Standards and Technology*. <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>

Metz, Cade (2011, 18 de octubre). «How Yahoo Spawned Hadoop, the Future of Big Data» [en línea]. *Wired*. <https://www.wired.com/2011/10/how-yahoo-spawned-hadoop>

O'Reilly Media (2012, 19 de enero). «Volume, Velocity, Variety: What You Need to Know About Big Data» [en línea]. *Forbes*. <https://www.forbes.com/sites/oreillymedia/2012/01/19/volume-velocity-variety-what-you-need-to-know-about-big-data>

Sarmenta, Luis F. G. (2001). *Volunteer computing* [en línea]. Cambridge: Massachusetts Institute of Technology. <http://people.csail.mit.edu/lfgs/papers/sarmenta-phd-mit2001.pdf>

Schollmeier, Rüdiger (2001). «A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications» [en línea]. *Proceedings First International Conference on Peer-to-Peer Computing* (págs. 0101). <https://doi.org/10.1109/P2P.2001.990434>

Nota: Todas las marcas registradas ® y licencias © pertenecen a sus respectivos propietarios. Todos los materiales, enlaces, imágenes, formatos, protocolos, marcas registradas, licencias e información propietaria utilizados en este documento son propiedad de sus respectivos autores o compañías, y se muestran con fines didácticos y sin ánimo de lucro, excepto aquellos que bajo licencias de uso o distribución libre han sido cedidos y/o publicados para tal fin (artículos 32-37 de la ley 23/2006, España).