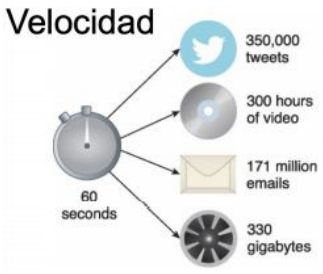
¿Qué es Big Data?

- Big Data se dedica al análisis, procesamiento y almacenamiento de una gran cantidad de datos provenientes de fuentes heterogéneas. Es una actualización de las técnicas tradicionales de análisis, procesamiento y almacenamiento por no ser suficientes.
- Big data Consists of datasets that grow so large that they become awkward to work with using on-hand DB Management tools - Consiste en conjuntos de datos que crecen tanto que resulta difícil trabajar con ellos utilizando herramientas de administración de bases de datos disponibles (Wikipedia).
- Big data is when the size of the data itself becomes part of the problem - Big data es cuando el tamaño de los datos en sí se convierte en parte del problema. (Mike Lukides, O'Reilly Radar)
- It's not just your "Big Data" problems, it's all about your BIG "data" Problems - No son solo sus problemas de "Big Data", se trata de sus GRANDES problemas de "datos" (Alexander Stojanovic, Hadoop Manager on Win Azure)

Campo de aplicación de Big Data

- Optimización de operaciones
- Identificación de nuevos mercados
- Predicción (Clima, desastres, bolsa de valores)
- Detección de fraudes
- Soporte a la toma de decisiones
- Descubrimientos científicos

Las 5 vs de BIG DATA

- Volumen: El tamaño de la data
- Velocidad: La velocidad con la que la data se genera
    - 

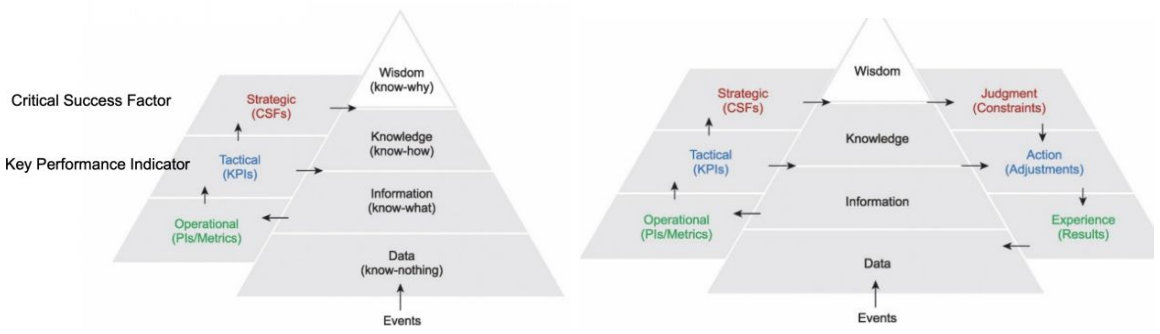
- Variedad: Los diferentes tipos de data
    - 

- Veracidad: La confiabilidad de los datos en términos de precisión
    - Twitter
- Valor: Big Data es útil cuando se puede convertir en valor (Lucrar de los datos)

Causas para la adopción de Big Data

- Nueva dinámica del mercado

    - Burbuja .com, recesión 2008.
    - Mantener rentabilidad y Reducir costos
    - Man/Tener nuevos y antiguos clientes, mediante nuevo productos/servicios o valor agregado al cliente
- Arquitectura del negocio



- Manejo de procesos de negocios
    - Descripción de cómo se realiza el trabajo.
    - Actividades del negocio y las relaciones con los actores responsables de ejecutarlas.
    - Procesos alineados a los objetivos del negocio
- TICs
    - Análisis de datos y ciencia de datos
    - Digitalización
    - Tecnología asequible y hardware básico
    - Medios de comunicación social
    - Comunidades y dispositivos hiperconectados/ Hyper-connected communities and devices
        - Se relaciona directo con la Internet de las cosas (IoT)
        - Permitir obtener información de todos los dispositivos posibles.
        - Utilizar como fuente de datos comunidades de información.
    - Computación en la nube
        - Capacidad de utilizar la nube para acceder a los datos
        - Utilizar la capacidad de procesamiento existente en aplicaciones almacenadas en la nube.
- Internet of Everything (IoE)
    - 14 billones
    - 2020: 32 billones

Características deseadas de un sistema de Big Data

- Robustez y tolerancia a fallos
    - *El sistema se comporta correctamente aunque algunos PCs se han caído.*
    - Compleja semántica y consistencia en base de datos distribuidas.
    - Los sistemas deben ser ”human-fault-tolerant”-tolerante a fallo humano
- Latencia baja en lecturas y escrituras
    - *Leer/Escribir mucha información en muy pocos segundos.*
    - Algunas aplicaciones requieren tiempo para propagar las actualizaciones en sus sistemas.
    - Se requiere leer rápidamente información sin comprometer la robustez del sistema

- Escalabilidad
    - *Capacidad de agregar nuevos datos o recursos sin comprometer el desempeño del sistema.*
    - La arquitectura Lambda es horizontalmente escalable a través de cada capa.
    - Se logra al añadir varias computadoras.
- Generalizable
    - *Puede soportar un número grande de aplicaciones.*
    - La arquitectura Lambda esta basada en función de todos los datos.
    - Los datos pueden ser de diferente tipo: financieros, social media, aplicaciones cientıficas.
- Extendible
    - *No reinventar la rueda cada vez que se quiera agregar una característica.*
    - *Agregar una funcionalidad requiere un costo mínimo de esfuerzo.*
    - A veces la inclusión de una nueva funcionalidad requiere de la migración de datos viejos en un nuevo formato.
    - *Capaz de migrar grandes cantidades de datos rápida y fácilmente.*
- Ad hoc queries
    - La posibilidad de *crear consultas específicas* para obtener información interesante.
- Mantenimiento Mínimo
- Depurable

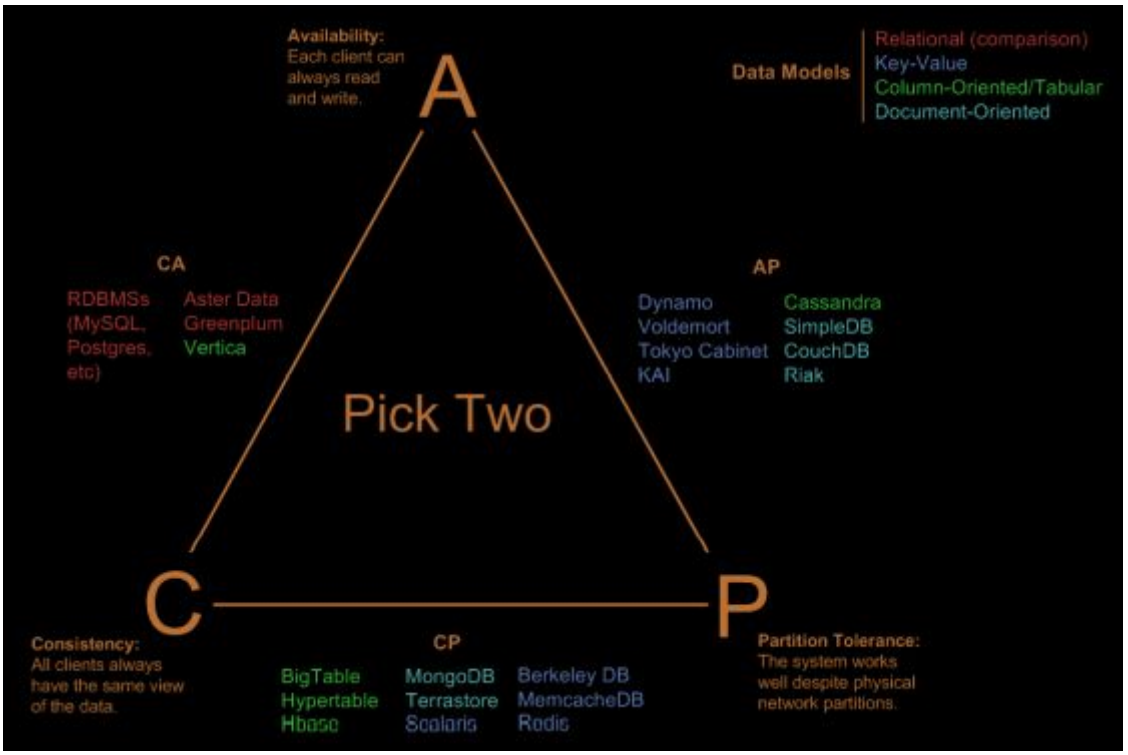
Soluciones de Big Data

- MapReduce: Framework de hardware distribuido (cluster/grids) que divide los problemas en subproblemas (Map) y luego se recopila las mini-respuestas (Reduce) para generar conclusiones. La solución más común es Hadoop. Modelo creado y promovido por Google.
- NoSQL Database
    - *Amplia clase de sistemas de gestión de bases de datos* que difieren del modelo clásico del sistema de gestión de bases de datos relacionales (RDBMS) en múltiples aspectos:
        - No usan SQL como principal lenguaje de consultas
        - Los datos almacenados *no requieren estructuras fijas como tablas*
        - *No garantizan ACID* (atomicidad, consistencia, aislamiento y durabilidad)
        - *Escalan bien horizontalmente* (ej: MongoDB, Cassandra, BigTable)
- Algoritmos genéticos
    - *Un algoritmo es una serie de pasos organizados que describen el proceso que se debe seguir, para dar solución a un problema específico.*
    - Con la inteligencia artificial, surgieron los algoritmos genéticos, inspirados en la evolución biológica
    - Evolucionan sometidos a mutaciones y recombinaciones genéticas.
- Recaptcha (Google)
    - reCAPTCHA es una extensión de la prueba CAPTCHA.
    - Reconocer texto presente en imágenes, usado para determinar si el usuario es o no humano.
    - Mejorar la digitalización de textos.
- Reconocimiento de patrones
    - *Ciencia que se ocupa de los procesos sobre:*

        - Ingenieria, computacion y matematicas
        - Relacionados con objetos fisicos o abstractos
        - Extracción de información que permita establecer propiedades de entre conjuntos de dichos objetos.
- NUI (Natural User Interface)
    - *Interfaz que permite interactuar con un sistema sin utilizar sistemas de mando o dispositivos de entrada de las GUI, usando en su lugar movimientos gestuales*.
    - Ej: Kinect. Reconocimiento de gestos y movimientos.

Teorema CAP

- Consistency (Consistencia), Availability (Disponibilidad) y Partition Tolerance (Tolerancia al Particionamiento)
- Nos dice que en un sistema distribuido de almacenamiento de datos no podemos garantizar consistencia y disponibilidad (para actualizaciones)
- Partición (queda separado en dos o mas islas).
- Depende de las exigencias del proyecto para saber que atributos de calidad es necesario y elegible.



El teorema es que solo puedes garantizar dos de estos tres atributos:
CP (Consistencia y Tolerancia al particionamiento):

- No disponibilidad
- No es elegible con clientes que requieren que el sistema esté disponible 100% del tiempo o muy cerca
- Se puede lograr en cierto nivel, pero el **sistema esta enfocado en aplicar los cambios de forma consistente aunque se pierda comunicación con algunos nodos**.

AP (Disponibilidad y Tolerancia al particionamiento):

- No garantiza datos iguales en todos los nodos todo el tiempo

- En este caso el **sistema siempre estará disponible para las peticiones aunque se pierda la comunicación entre los nodos**.

CA (Consistencia y disponibilidad):

- No particionado de los datos, porque se **garantiza que los datos siempre son iguales y el sistema estará disponible respondiendo todas las peticiones**.
- Por ejemplo, los sistemas de bases de datos relacionales (SQL) son CA porque todas las escrituras y lecturas se hacen sobre la misma copia de los datos.

Presentaciones

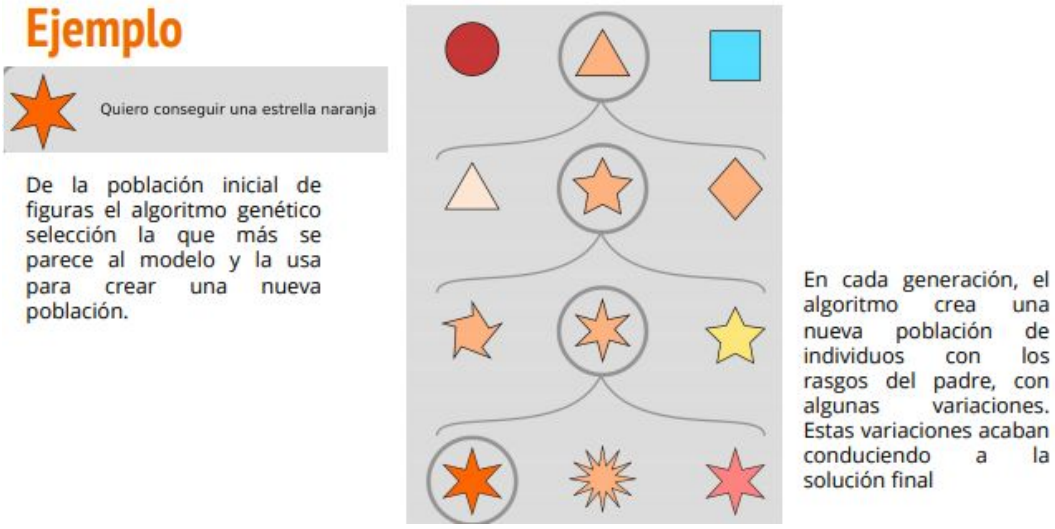
- Cassandra: Es clave valor, CAP es AP
- CouchDB: Es documental, CAP es AP
- Mongo: Es documental, CAP depende contexto, mayoritariamente CP
- Firebase: Es documental organizado en colecciones, CAP es CP
- REDIS: Es clave valor, Redis es CP xq deja de estar disponible en particiones minoritarias

Algoritmo Genético

Inspiran en la evolución natural para solucionar problemas de optimización que de otra forma serían difíciles para un diseñador humano. Se tiene un conjunto de soluciones al problema a resolver. Se llama población al conjunto de soluciones e individuo a cada una de las soluciones. **El algoritmo evalúa cada una de las soluciones y selecciona las que mejor resuelvan el problema**

Algoritmo: Serie de pasos que describen el proceso de búsqueda de una solución a un problema concreto.

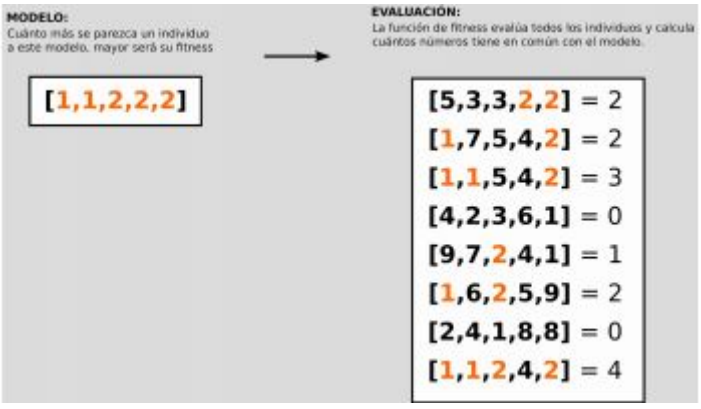
Algoritmo genético: Usan mecanismos que simulan los de la evolución de las especies de la biología para formular dichos pasos. Es una **técnica de IA** inspirada en la idea de que el que **sobrevive es el que está mejor adaptado al medio** (teoría evolución de Darwin, combina esa idea de la evolución con la genética). Son **algoritmos de optimización búsqueda y aprendizaje** inspirados en los procesos de Evolución Natural y Genética.



Pasos para construir un Algoritmo genético

- Inicialización
- Evaluación/Función de Fitness
- Selección
- Reproducción
- Crossover
- Mutación

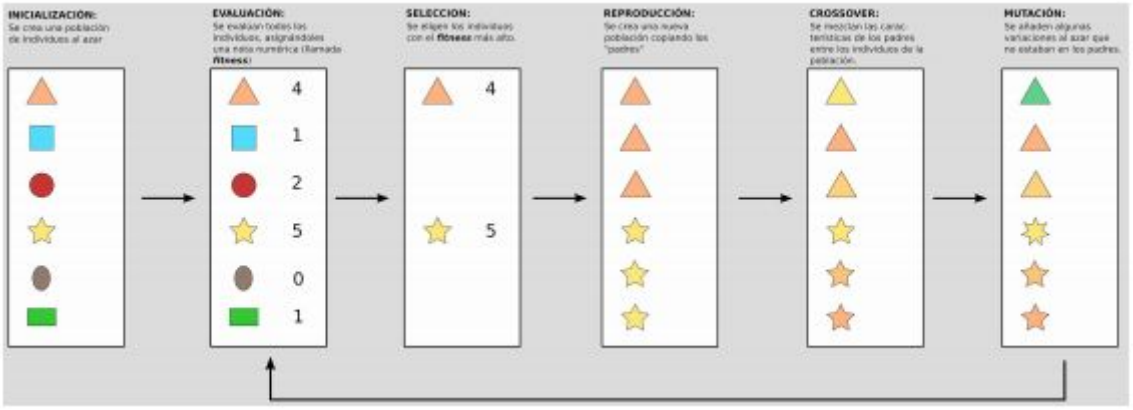
1. Inicialización: La población inicial puede generarse al azar, pero debe ser grande y diversa para que durante la evaluación los individuos muestran un fitness distinto. Si todos los individuos tienen el mismo fitness, el programa no selecciona bien los mejores para su reproducción.
2. Evaluación/Función de Fitness: La Función de Fitness evalúa a cada uno de los individuos de la población y les asigna un valor numérico según su calidad. **Este valor numérico se llama fitness.**



3. Selección/Reproducción: Varias formas de seleccionar los individuos con el fitness más elevado para crear una nueva población. Algunas de las más utilizadas son:
   Selección proporcional: cuanto más alto sea el fitness de un individuo, mayor será la probabilidad de que pase a la siguiente generación.
   Selección por torneo: se eligen varios individuos al azar de la población y se compara su fitness. El individuo con el fitness más alto pasará a la siguiente generación. Hay ocasiones en que es interesante escoger soluciones que no son tan

buenas para poder mantener una buena variabilidad genética entre los individuos. Así al mezclar el material genético se exploran soluciones muy distintas.
Selección por rango: los individuos se ordenan en función de su fitness, y la probabilidad que tienen de reproducirse va según su posición.

4. Crossover/Mutación: Una vez se han seleccionado los mejores individuos, el **Crossover consiste en mezclar el material genético de estos para crear los nuevos individuos.** La forma más sencilla es mediante un One-point Crossover, consiste en elegir un punto al azar de los dos individuos e intercambiar el material genético a partir de esta posición. El crossover no ayuda a encontrar nuevas soluciones al problema. Es por eso que hay que mutar el material genético de los nuevos individuos, es decir: añadir pequeñas variaciones al azar.



Aplicación

- Diseño automatizado, incluyendo investigación en diseño de materiales y diseño multiobjetivo de componentes automovilísticos: mejor comportamiento ante choques, ahorros de peso, mejora de aerodinámica, etc.
- Diseño automatizado de equipamiento industrial.
- Diseño automatizado de sistemas de comercio en el sector financiero. ● Construcción de árboles filogenéticos.
- Optimización de carga de contenedores.
- Diseño de sistemas de distribución de aguas.
- Diseño de topologías de circuitos impresos.
- Diseño de topologías de redes computacionales.
- En teoría de juegos, resolución de equilibrios.

Relación con Big Data

- El principal problema de Big Data es la pérdida de rendimiento de los algoritmos de aprendizaje automático y minería de datos. Porque se tiene una gran cantidad de características
- Aplicación de AG relacionada con Big Data se da en el procesamiento de grandes cantidades de datos. Combina con técnicas de clustering para realizar el eficiente procesamiento de los datos. O combinación de técnicas de AG con un procesamiento en paralelo
- Uso de Algoritmos Genéticos para:
    - Clasificar los datos de manera eficiente.
    - Extraer la información relevante de la gran cantidad de datos como paso previo para luego aplicar el algoritmo de clasificación.

La implementación de un algoritmo genético como paso previo resulta eficiente para resolver el problema de selección de características

Ventajas:

- **Mejor que la IA convencional por ser más robusta**, no se rompen fácilmente incluso si las entradas cambian levemente o en presencia de ruido razonable. Además, al buscar en un gran espacio de estado multimodal o superficie n-dimensional, un algoritmo genético puede ofrecer beneficios significativos sobre una búsqueda más típica de técnicas de optimización.

Desventajas

- Función evaluación costosa en tiempo y recursos en problemas de alta complejidad.
- Algoritmo podría no converger en una solución óptima o terminar en una convergencia prematura con resultados no satisfactorios
- Mala escalabilidad con la complejidad.
- La "mejor" solución es solo en comparación a otras soluciones. No se tiene un criterio claro de cuándo detenerse porque no se cuenta con una solución específica.
- No es recomendable usarlos para problemas que convergen en soluciones simples como Correcto/Incorrecto ya que el algoritmo difícilmente convergerá y el resultado será tan válido como escogerlo al azar.
- El diseño, la creación de la función de aptitud y la selección de los criterios de mutación entre otros, necesitan de cierta pericia y conocimiento del problema para obtener buenos resultados.

CASSANDRA

- Base de datos NoSQL distribuida que maneja grandes volúmenes de datos.
- El objetivo principal es la **escalabilidad lineal y la disponibilidad**.
- Soporte robusto para múltiples centros de datos, con la replicación asincrónica sin necesidad de un servidor maestro, que permiten operaciones de baja latencia para todos los clientes.

Características

- Arquitectura Estable: Diseño masterless, donde todos los nodos son iguales. Ofrece simplicidad operativa y fácil escalabilidad horizontal.
- Diseño activo de principio a fin: Escribir y leer en todos los nodos.
- Rendimiento a escala lineal: Puede añadir nodos sin frenar el ritmo, produce aumentos en el rendimiento.
- Disponibilidad continua: elimina los puntos únicos de fallo y proporciona un tiempo de actividad constante.
- Detección de fallos y recuperación transparente: fácil restauración en eliminación de nodos.
- Modelo de datos flexible y dinámico: soporta tipos de datos modernos para lectura y escritura rápida.
- Protección de datos sólida: diseño de registro de confirmación evita la pérdida de datos y construye copias de seguridad.
- Consistencia de los datos sintonizables: consistencia de los datos en un clúster ampliamente distribuido.
- Replicación de datos multi-centro: centro de datos transversal (diferentes zonas geográficas) que recibe el el apoyo de múltiples zonas de disponibilidad en la nube tanto para escritura como para lectura.

- Compresión de datos: garantiza la comprensión de datos hasta un 80% sin que ello suponga un gasto de recursos.
- CQL (Lenguaje de Consulta Cassandra): similar a SQL que consigue que la transición desde una base de datos relacional sea muy sencilla.
- En Cassandra los datos están desnormalizados de manera que el **concepto de joins o subqueries no existe.** Interacción mediante shell con cqlshell, o GUI como DevCenter o por drivers soportados para múltiples lenguajes de programación.

Cuando elegir Cassandra.

- Cuando se quiere obtener un escalamiento lineal, un nivel alto de disponibilidad y cortos tiempos de respuesta.
- Debido a las limitaciones que tiene en filtros y ordenaciones, el conocimiento de las consultas a realizar sobre la base de datos tiene un gran impacto en el modelo a definir.

Modelo de Datos.

- Combina propiedades de una base de datos clave-valor y una orientada a columnas.
- Importante definir clave de partición de data, ya que Cassandra usa esta para distribuir los datos a lo largo del cluster.
- Guiarse en diseño por el patrón de acceso a los datos, hay que hacer un análisis de las queries a ejecutar para diseñar un modelo eficiente.

Cassandra da prioridad a la escalabilidad, velocidad y alta disponibilidad por encima de la consistencia.

COUCHDB

Gestor de base de datos NoSQL de código abierto. Desarrollado por Apache Software Foundation en 2005.

AP: DISPONIBLE Y PARTICIONABLE

Compatible con los sistemas operativos Linux, Unix, MacOS y Windows. Escrito en el lenguaje de programación Erlang.

JSON – almacenar datos. API JSON. JavaScript – lenguaje de consulta mediante MapReduce.

HTTP – solicitudes que se emiten desde el navegador.

CouchDB guarda los datos en forma de documentos (almacenados en JSON).

- **_id:** Usado para que CouchDB lo distinga de otros documentos y pueda recuperarlos.
- **_rev:** Controlador de versiones.
- **Otros campos**: Usan expresiones JSON válidas como arrays de strings.

MONGO

Tipos de NOSQL: Bases de datos Documentales, Orientadas a grafos, Bases de datos Clave/Valor, Bases de datos Multivalor, Bases de datos Tabulares, Bases de datos de Arrays.

Sistema de base de datos NoSQL **orientado a documentos (**no guardar datos en registros sino en documentos**),** almacenados en BSON, de código abierto y escrito en C++, multiplataforma.

Si se necesita realizar consultas de agregación MongoDB tiene un Framework para realizar consultas llamado Aggregation Framework.

Algunas aplicaciones que usan MongoDB: WindyGrid, Bosch, AIR FRANCE.

- garantiza operaciones atómicas a nivel de documento, no usan joins
- Balance entre rendimiento y funcionalidad
- Alta velocidad
- *Consultas ad hoc*: se puede realizar todo tipo de consultas, p.e. hacer búsqueda por campos, consultas de rangos y expresiones regulares.
- *Indexación*: similar a bd relacionales, con la diferencia de que cualquier campo documentado puede ser indexado y añadir múltiples índices secundarios.
- *Replicación*: soporta replicación primario-secundario
- *Balanceo de carga*: ejecutarse de manera simultánea en múltiples servidores
- *Ejecución de JavaScript del lado del servidor*: consultas utilizando JavaScript, son enviadas directamente a la base de datos para ser ejecutadas.

CAP:

C: Muy consistente cuando usa una sola conexión o el nivel de preocupación de escritura / lectura correcto (cuesta velocidad de ejecución).

A: Alta disponibilidad a través de Replica-Sets. Tan pronto como la primaria deje de funcionar o deje de estar disponible, las secundarias determinarán una nueva primaria para volver a estar disponible.

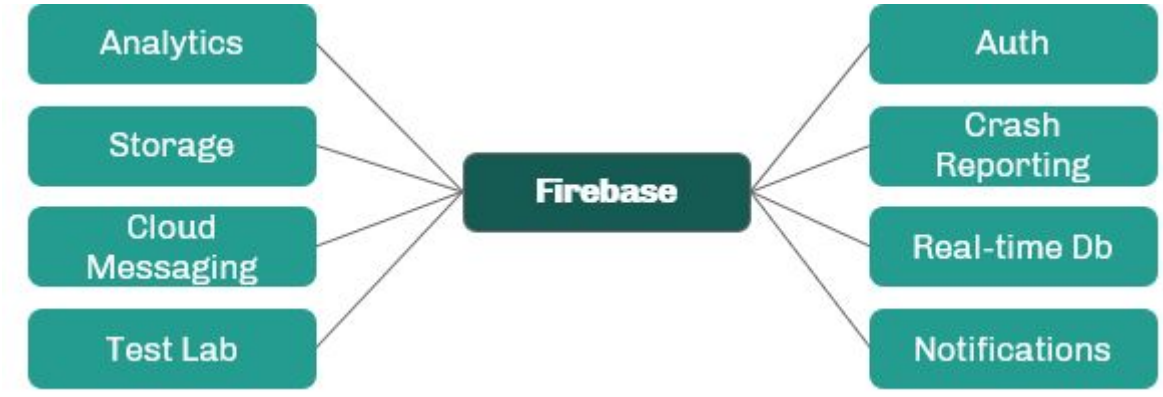
P: tolerancia de partición: siempre que más de la mitad de los servidores de un conjunto de réplicas están conectados entre sí, se puede elegir un nuevo primario.

No se puede simplemente decir que MongoDB es CP / AP / CA, porque en realidad es una compensación entre C, A y P, dependiendo de la configuración de la base de datos / controlador y el tipo de criterio.

Para un sistema distribuido como Mongodb siempre se construye con tolerancia a particiones, por lo tanto se puede decir que si hay una partición de red y si se desea que su sistema siga funcionando, puede proporcionar Disponibilidad o Consistencia y no ambas.

Firebase

- Es todo un backend, actualmente tiene 8 tecnologías que mejora la experiencia de desarrollo de aplicaciones.



- FireStore es una base de datos NoSql de tipo documentos de Firebase.
- Online y Offline - Proporciona sincronización entre dispositivos conectados y está disponible cuando no hay conectividad de red.
- **En CAP es CP**
- Cloud Firestore es una base de datos NoSQL alojada en la nube, flexible y escalable para la programación en servidores, dispositivos móviles y la web desde Firebase y

Google Cloud Platform. Almacenar los datos en documentos que contiene campos que se asignan a valores. Los documentos se organizan en colecciones, que son contenedores para los documentos y se pueden usar para organizar los datos y compilar consultas.

- Las consultas son expresivas, eficientes y flexibles.
- Los documentos admiten diferentes tipos de datos, desde strings hasta objetos anidados complejos o de subcolecciones dentro de documentos.
- Firebase Realtime Database, mantiene la sincronización de los datos entre apps cliente a través de agentes de escucha en tiempo real y ofrece asistencia sin conexión.
- Recomendable en proyectos con tiempos de desarrollo limitados.
- Recomendable cuando se requiera construir aplicaciones online por sobre los offline.
- Recomendable en proyectos con manejo de dispositivos inteligentes con mínimos requerimientos de uso.
- Recomendable en proyectos con correcta planificación de costes..

REDIS

Motor de base de datos en memoria, basado en el almacenamiento en tablas de hashes (clave/valor) pero que opcionalmente puede ser usada como una base de datos durable o persistente.

Redis es CP porque deja de estar disponible en particiones minoritarias. Tenga en cuenta que seguirá estando disponible en la partición mayoritaria.

Ventajas:

Una velocidad muy alta, gracias a su almacenamiento en memoria

- Posibilidad de persistir datos en disco para recuperación ante fallas
- Fácil configuración
- Alta disponibilidad
- Curva de aprendizaje baja
- Una variedad de tipos de datos

Desventajas:

- El método de persistencia RDB consume mucho I/O (escritura en disco)
- Todos los datos trabajados deben encajar en la memoria (en caso de no usar persistencia física)