PEC1

SISTEMAS DISTRIBUIDOS

Introducción y conceptos generales

TABLA DE CONTENIDO

Pregunta 1 2
Pregunta 2 3
Pregunta 3 5
Pregunta 4 6
Pregunta 5 8
Referencias 12

PREGUNTA 1

Describe el concepto de transparencia en sistemas distribuidos.

Explica los modelos de transparencia (ubicación, replicación, fallos) relacionado con la escalabilidad de un sistema distribuido.

La transparencia se define como la ocultación de la separación de los componentes del sistema distribuido a ojos de los usuarios y de los programadores de aplicaciones, dando una imagen de un "todo" en lugar de diversos componentes independientes.

La transparencia cobra vital importancia en el diseño del software de sistema.

Basándonos en el Manual de Referencia ANSA y la Organización Internacional para la Normalización del modelo de referencia para el procesamiento distribuido abierto se pueden identificar ocho formas de transparencia, entre las cuales se encuentran los tres modelos indicados en el enunciado:

- **Transparencia de acceso**: permite que el acceso a los recursos locales y remotos se haga mediante operaciones idénticas.
- **Transparencia de ubicación**: permite que al acceso a los recursos se haga sin necesidad de conocer su ubicación física o de red.
- **Transparencia de concurrencia**: permite a varios procesos operar concurrentemente usando recursos compartidos sin que se interfieran entre ellos.
- **Transparencia de replicación**: permite múltiples instancias de los recursos con el fin de aumentar la fiabilidad y el rendimiento sin que los usuarios o los programadores tengan conocimiento de dichas réplicas.
- **Transparencia de fallos**: permite la ocultación de errores, permitiendo a los usuarios y a los programas que completen sus tareas a pesar de los fallos de los componentes de hardware o software.
- **Transparencia de movilidad:** permite el movimiento de recursos y clientes dentro de un sistema sin que la actividad de usuarios o programas se vea afectada.
- **Transparencia de rendimiento**: permite al sistema reconfigurarse para mejorar el rendimiento ante las variaciones de cargas.
- **Transparencia de escala**: permite al sistema y a las aplicaciones expandirse en escala sin modificar la estructura del sistema o los algoritmos de la aplicación.

Un sistema se describe como escalable si su eficacia no se ve comprometida como consecuencia del aumento del número de recursos y usuarios.

El diseño de sistemas distribuidos escalables presenta los siguientes retos:

- Controlar el coste de los recursos físicos: ya que a medida que la demanda de un recurso crece debería ser posible extender el sistema, añadiendo componentes físicos.
- Controlar la pérdida de rendimiento: un aumento de tamaño se traduce en una pérdida de rendimiento, se considera que un sistema es escalable cuando esta pérdida no supera $O(\log n)$.
- Evitar que se agoten los recursos del software: haciendo una buena previsión para que estos puedan cubrir la demanda que tendrán.
- Evitar los cuellos de botella de rendimiento: descentralizando los algoritmos para evitar que un aumento en la demanda de recursos genere esta situación.

Por lo tanto y sabiendo esto, podemos redefinir las tres formas de transparencia indicadas en el enunciado desde el punto de vista de la escalabilidad:

- **Transparencia de ubicación**: como dijimos antes, hace transparente la ubicación de los recursos, lo cual facilita que el sistema se pueda escalar cuando sea necesario aumentando recursos físicos y modificando la estructura del sistema sin que esto afecte al usuario final.
- **Transparencia de replicación**: la replicación de los recursos ayuda a evitar los cuellos de botella cuando se escala un sistema, ya que estos van a estar descentralizados y el sistema va a permitir múltiples fuentes para acceder a ellos.
- **Transparencia de fallos**: la ocultación de los errores es el soporte vital para controlar la pérdida de rendimiento del sistema que se puede dar cuando aumenta su uso. También se relaciona directamente con los dos tipos de transparencia que hemos relacionado con la escalabilidad, ya que va a ocultar todo el proceso que puede haber detrás de un cuello de botella o de la caída de algún recurso físico.

(Coulouris, Dollimore, Kindberg, & Blair, 1.5.7 Challenges: Transparency, 2012)

PREGUNTA 2

Explica los principales modelos de fallos en los sistemas distribuidos.

Explica las distintas técnicas para lograr transparencia frente a fallos.

En un sistema distribuido puede fallar tanto los procesos como los canales de comunicación, es decir, puede no obtenerse de ellos el comportamiento esperado.

El modelo de fallos define cómo se pueden producir los fallos y así poder entender los efectos de los mismos.

Hadzilacos y Toueg (1994) proporcionaron una distinción entre fallos de procesos y de los canales de comunicación y los principales modelos de fallos en sistemas distribuidos son:

- Fallos por omisión: cuando un proceso o un canal de comunicación no realiza las acciones que se supone que tenía que hacer. Dentro de estos encontramos:
    - Proceso: el principal fallo por omisión de proceso es cuando este se detiene (crash). Puede haber servicios que toleren estos fallos siempre y cuando el proceso se detenga limpiamente (o bien funciona correctamente o bien se detiene). Otros procesos pueden detectar el colapso debido al fallo repetido del proceso al responder a los mensajes de invocación, aunque en un sistema asíncrono esto podría representar simplemente lentitud y no una detención del proceso. En un sistema síncrono otros procesos pueden usar un tiempo determinado para detectar si un proceso se ha detenido y bloquearlo (fail-stop).
    - Comunicación: el canal de comunicación produce un fallo de omisión si no transporta un mensaje de un buffer a otro (caída de mensajes) y suele obedecer a una falta de espacio en el buffer o a un error de transmisión de la red

Los fallos pueden clasificarse también según su gravedad. Los fallos descritos hasta ahora son benignos (como la mayoría de los fallos en sistemas distribuidos).

- Fallos arbitrarios: describe la peor semántica de fallo posible, en el que puede ocurrir cualquier error.

    - Proceso: el fallo arbitrario de un proceso es aquel en el que se omiten arbitrariamente pasos de procesamiento o bien se añaden pasos no previstos. No puede detectarse mediante la respuesta a las invocaciones ya que las puede omitir arbitrariamente.
    - Comunicación: el fallo arbitrario en los canales de comunicación es raro ya que el software de comunicación es capaz de reconocerlos y rechazar aquellos mensajes defectuosos. Consiste en mensajes erróneos, corrompidos, entregas de mensajes inexistentes o duplicados. Usar checksums o números de secuencia de mensajes permite detectar este tipo de fallos.

- Fallos de tiempo: aplicables en sistemas distribuidos síncronos que establecen límites de tiempo para el tiempo de ejecución de procesos y el tiempo de entrega de mensajes. En un sistema distribuido asíncrono el servidor puede responder con lentitud, pero no se puede considerar un fallo de tiempo ya que no se ha ofrecido ninguna garantía. La mayoría de los sistemas operativos de uso general no tienen que cumplir con las restricciones de tiempo real. El tiempo es particularmente pertinente en el caso de equipos multimedia con audio y video con un gran número de intercambio de mensajes lo cual plantea exigencias de sincronización a nivel de procesos y de comunicación.

En cuanto a las técnicas para lograr transparencia ante los fallos se ha comentado por encima alguna dentro de los apartados, pero se pueden resumir a grandes rasgos en dos categorías:

- Enmascaramiento de fallos: el conocimiento de las características de los fallos de un componente permite diseñar servicios para enmascararlos. Un servicio enmascara un fallo ya sea ocultándolo por completo o convirtiéndolo en un tipo de fallo más aceptable (por ejemplo, mediante los checksums que comentamos se puede lograr transformar un fallo arbitrario en un fallo por omisión). Los fallos por omisión pueden ocultarse mediante un protocolo que retransmite los mensajes que no llegan a su destino y la replicación también es útil para lograr el enmascaramiento. Con los fallos de proceso es posible reemplazar y restaurar el proceso logrando así que no se quede detenido.
- Fiabilidad de la comunicación uno a uno: es posible utilizar un canal de comunicación para construir un servicio de comunicación que enmascare fallos. Una comunicación fiable se basa en la validez (el hecho de que el mensaje realmente se entregue) y la integridad (que el mensaje sea el original y que no se envíe por duplicado). Mediante medidas de secuenciación de mensajes y seguridad se puede evitar la falta de integridad de estos ante ataques.

(Coulouris, Dollimore, Kindberg, & Blair, 2.4.2 Fundamental models: Failure model, 2012)

(Coulouris, Dollimore, Kindberg, & Blair, 1.5.5 Challenges: Failure handling, 2012)

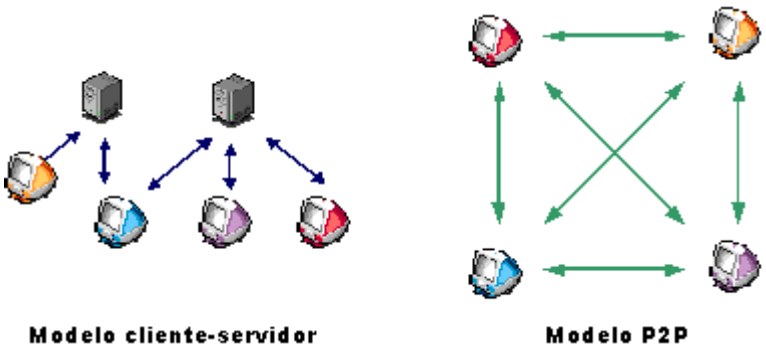
PREGUNTA 3

Compara el modelo peer to peer con el modelo cliente-servidor escalonado utilizado en la Nube.

Compara utilizando conceptos como la simplicidad de diseño, la tolerancia a los fallos, la heterogeneidad y la escalabilidad.

La arquitectura de un sistema es su estructura en términos de componentes y las relaciones que hay entre ellos. Cada modelo tiene como objetivo asegurar que dicha estructura satisfaga las demandas presentes y futuras que se le presentarán.

Los modelos p2p y cliente-servidor son dos entre los modelos de arquitectura de sistemas distribuidos que existen actualmente. Dentro del modelo cliente servidor tenemos el modelo N-tier, donde N se refiere al número de capas.



- Modelo peer to peer: este sistema se basa en tratar todos los procesos implicados como iguales, donde no hay roles y todos interactúan cooperativamente. Su objetivo es explotar los recursos de un gran numero de computadores participantes para el cumplimiento de una tarea o actividad determinada.
    - Heterogeneidad: este sistema distribuye la base de datos de la aplicación y las cargas de almacenamiento, procesamiento y comunicación entre muchos computadores y enlaces de red, lo cual le proporciona una gran heterogeneidad.
    - Tolerancia a los fallos: cada objeto se distribuye entre muchos computadores y que este se replique distribuyendo la carga hace que la tolerancia a fallos sea mayor al aportar alternativas a fallos físicos y evitando los cuellos de botella.
    - Escalabilidad: la escalabilidad no representa un problema para esta arquitectura, ya que es la piedra angular que la sustenta. La clave principal de este modelo es usar la red y los recursos informáticos de propiedad de los usuarios de un servicio para apoyarlo. Esto provoca que los recursos disponibles para ejecutar el servicio crezcan al mismo ritmo que su número de usuarios.
    - Simplicidad de diseño: la necesidad de colocar objetos individuales, recuperarlos y mantener réplicas entre muchos computadores hace que esta arquitectura sea más compleja que la arquitectura cliente-servidor.

- Modelo cliente-servidor: ofrece un enfoque directo y simple para el intercambio de recursos basado en la centralización de la provisión y gestión de servicios: un servicio se sitúa en una dirección que depende de la capacidad del servidor que lo aloja y de su

ancho de banda. Actualmente sigue siendo la arquitectura más utilizada. Los procesos toman roles (cliente o servidor), en particular, los procesos clientes interactúan con los procesos individuales de los servidores en computadoras anfitrionas potencialmente separadas para acceder a los recursos compartidos que administran.

- Heterogeneidad: este modelo es muy poco heterogéneo ya que los servidores suelen tener unas características parecidas en cuanto a sistemas operativos, hardware y lenguajes de programación.
- Tolerancia a los fallos: en este modelo se pueden implementar más fácilmente las medidas de seguridad y enmascaramiento para tratar con los fallos, en contrapartida se requiere de una acción extra de replicación de datos para poder tratar fallos físicos, lo cual requiere una inversión.
- Escalabilidad: el modelo es escalable, como hemos dicho, previa inversión en equipos físicos. Los servidores pueden a su vez ser clientes de otros servidores, e incrementar el número de capas del modelo N-tier, distribuyendo en parte su función, pero sigue siendo menos escalable que el modelo p2p.
- Simplicidad de diseño: es sin duda el sistema más sencillo de implementar y por ello el más usado. Los roles están muy marcados y claros, las funciones que realiza cada equipo son muy claras y no se ha de implementar toda la logística que tiene detrás el p2p.

(Coulouris, Dollimore, Kindberg, & Blair, 2.3.1 Architectural models: Architectural elements, 2012)

(Coulouris, Dollimore, Kindberg, & Blair, 1.5 Challenges, 2012)

PREGUNTA 4

Explica el modelo de seguridad del código móvil en arquitecturas web.

Describe las técnicas para hacer frente a las amenazas en este entorno.

El código móvil es el código de programa que puede transferirse entre equipos y ejecutarse en el destino. La inclusión de programas JavaScript en páginas web cargadas en los navegadores de los clientes son un ejemplo de ello.

La seguridad de los recursos de información en los sistemas distribuidos tiene tres componentes: confidencialidad (protección de la divulgación de personas no autorizadas), integridad (protección contra la alteración o corrupción) y disponibilidad (protección al acceso a los recursos).

El desafío es enviar información sensible en un mensaje a través de una red de forma segura. Para ello no sólo hay que ocultar el contenido de los mensajes, sino también conocer con certeza la identidad de la fuente de los mensajes. Por otro lado, el segundo desafío es identificar correctamente a un usuario remoto. Existen técnicas de cifrado desarrolladas para este fin.

Sin embargo, los siguientes desafíos de seguridad no se han podido terminar de resolver del todo:

- Ataques de denegación de servicio: operación maliciosa que consiste en interrumpir un servicio mediante un número desproporcionado de solicitudes inútiles. Actualmente hay acciones legales para intentar contrarrestarlos, pero no es una solución al problema. También se están desarrollando mejoras en la gestión de redes.

- Seguridad del código móvil: uso malicioso de este para, mediante el envío de un programa ejecutable, acceder a los recursos locales o actuar como parte de un ataque de denegación de servicio.

Para proteger a los usuarios del código móvil no confiable, la mayoría de los navegadores especifican que los applets no puedan acceder a los servicios locales del usuario, así como a impresoras o puertos de red.

Algunas aplicaciones son capaces de asumir varios niveles de confianza en el código descargado.

Las máquinas virtuales de Java toman además dos medidas de seguridad adicionales:

- Las clases descargadas se almacenan por separado de las clases locales, evitando que se sustituyan por versiones falsas.
- Se comprueba la validez de códigos de bytes. El código válido está compuesto por un conjunto de instrucciones específicas de la máquina virtual de Java. Estas se comprueban para asegurar que no produzcan errores como, por ejemplo, el acceso a direcciones de memoria ilegales.

A pesar de incluir mecanismos de verificación de tipos y validación de códigos, los sistemas de códigos móviles no producen el mismo nivel de confianza que los usados para proteger canales de comunicación e interfaces, ya que la construcción de un entorno de ejecución de programas es una puerta abierta a los errores humanos.

Un enfoque alternativo es el dado por Volpano y Smith que señalan que mediante pruebas de que el comportamiento del código es sólido se podría ofrecer una mejor solución.

(Coulouris, Dollimore, Kindberg, & Blair, 1.5.1 Challenges: Heterogeneity, 2012)

(Coulouris, Dollimore, Kindberg, & Blair, 1.5.3 Challenges: Security, 2012)

(Coulouris, Dollimore, Kindberg, & Blair, 2.3.1 Architectural models: Architectural elements, 2012)

(Coulouris, Dollimore, Kindberg, & Blair, 11.1.1 Introduction: Threats and attacks, 2012)

PREGUNTA 5

Lee el siguiente artículo y responde las preguntas "Towards Federated Learning at scale: system design":

a) **Resume el artículo y esboza los principales beneficios y problemas de las arquitecturas de aprendizaje federado.**

RESUMEN

El artículo expone un nuevo sistema para tratar datos basado en los sistemas distribuidos y el machine learning.

Su enfoque es un aprendizaje automático descentralizado que permita la capacitación de un conjunto de datos que residen en dispositivos como los teléfonos móviles.

El sistema permite entrenar una red neuronal profunda usando TensorFlow sobre los datos almacenados en el teléfono que nunca saldrán del dispositivo.

Se construye un modelo global con el promedio federado que es llevado a los teléfonos para su inferencia. El servidor selecciona un conjunto de dispositivos, envía el modelo global, los dispositivos llevan a cabo un cálculo y envían una actualización al servidor, si se han recibido suficientes reportes la ronda termina y el servidor actualiza el modelo global.

El proceso se divide en cuatro rondas sincrónicas: las dos primeras son una fase de preparación en la que se establece la información compartida encriptada; la tercera ronda es una fase de commit en la que se cargan las actualizaciones y en la que el servidor las acumula; la última ronda es la de finalización en la que los dispositivos facilitan la decodificación por parte del servidor de la información compartida

El sistema está preparado para escalarse automáticamente en función del número y tamaño de las poblaciones de dispositivos.

También está formado por una serie de actores definidos con unas tareas específicas con la finalidad de coordinar, supervisar y garantizar el éxito del proceso.

Por último, se usan técnicas como la Agregación Segura para mantener la confidencialidad o el análisis de los datos del servidor para evitar una afectación en la utilidad del dispositivo para el usuario.

BENEFICIOS

El principal y mas relevante beneficio de esta tecnología es la privacidad de los datos, ya que estos no han de tratarse ni leerse, son los propios dispositivos particulares quienes lo hacen y lo único que se comparte son los algoritmos con las conclusiones.

El banco de datos al que accede el sistema es mucho mayor que el que pueda contener una empresa de análisis en sus instalaciones, resulta más barato de obtener y de mantener al desaparecer la necesidad de almacenamiento de datos.

Los datos siempre estarán mucho más actualizados que los que hay en un centro de datos, ya que son los que están en uso en los teléfonos.

PROBLEMAS

Los retos a los que se enfrenta el aprendizaje federado son por un lado los técnicos descritos en el artículo: la necesidad de refinar algoritmos que optimicen las conclusiones extraídas a partir de las aplicaciones más usadas pero sin comprometer la fiabilidad, solventar el hecho de que muchos más teléfonos de los que se contaba no terminaban las rondas, optimizar también los algoritmos que se encargan de dar respuesta en estos supuestos, dinamizándolos en lugar de tener unos tiempos estáticos o valorar otras clases de algoritmos más eficientes en cuanto al paralelismo en su ejecución.

Requiere que se esté constantemente al día en cuanto a seguridad informática, tanto que para terceros no accedan a los datos de los componentes de la red como para que nadie pueda manipular los datos con los que se trabaja y puedan dar unos falsos resultados. Hasta ahora esto se lleva a cabo mediante agregación segura y encriptación, pero es un campo en el que se deberá invertir unos recursos que un centro de datos al uso no debe destinar (como mínimo a esa escala).

Por otro lado, creo que el mayor problema al que se enfrenta este tipo de tecnología (y no solo el aprendizaje federado, sino todo aquél que requiera del uso de datos) es la desconfianza global que hay con respecto a la privacidad. A pesar de que este es uno de los sistemas que más respeta los datos personales el público no puede evitar pensar que si se accede a su teléfono no se va a respetar y, en resumen, eso genera una gran desconfianza por parte del público.

b) Relaciona los conceptos de los modelos de arquitectura, seguridad, transparencia, modelos de fallo y escalabilidad con estas arquitecturas.

El **modelo de arquitectura** del aprendizaje federado es una mezcla entre las arquitecturas web peer to peer y cliente servidor, que definimos y tratamos en la pregunta 3.

Por un lado, tenemos el hecho de que todos los dispositivos toman parte del trabajo, como iguales, del mismo modo que se hace con la arquitectura P2P. Por otro lado, tenemos un servidor que se encarga en última instancia de tomar un rol de coordinador, orquestando todo el proceso de rondas de aprendizaje y almacenando el funcionamiento de estas para poder analizarlo y mejorarlo.

El **modelo de seguridad** es exactamente el mismo que el que se ha visto para los sistemas distribuidos (descrito en la pregunta 4 brevemente): confidencialidad, integridad y disponibilidad.

La confidencialidad se trata mediante la agregación segura y la encriptación, también está el hecho de que los datos no salen del dispositivo. La integridad de los datos se basa en saber que realmente el dispositivo que está participando es quien dice ser, el reconocimiento del usuario no por sus datos personales sino por el mecanismo de certificación remota de Android hace que esto sea posible sin vulnerar la privacidad. Por último, la disponibilidad tiene una serie de algoritmos detrás que son los encargados de que sólo se usen dispositivos que estén en reposo sin perjudicar la usabilidad de estos y que todo un sistema de tiempos haga que el sistema se adapte a la cantidad de dispositivos que se pueden conectar a las rondas (lo hace escalable).

La **transparencia** viene unida al punto anterior, ya que los algoritmos comentados con anterioridad permiten que el usuario no sea consciente de la participación de su dispositivo en el sistema. Se procura que el dispositivo no esté en uso, que se esté cargando y que no dependa

de una red de datos móvil. Si alguna de estas condiciones no se cumple el dispositivo no entra en la ronda (o sale de ella en caso de que ya estuviera dentro).

El **modelo de fallos** de nuevo reúne lo mejor de cada arquitectura, por un lado, la distribución de las tareas hace que si falla alguno de los dispositivos no se aborte la ronda y por otro, la existencia de un servidor que coordina todo el proceso hace que se puedan implementar algoritmos con roles concretos (coordinador, selectores, agregadores y agregador maestro) que se encargan de enmascarar los fallos reiniciando o replicando los actores o procesos que sean necesarios en función del fallo para que el sistema siga avanzando independientemente de lo que ocurra.

La **escalabilidad** es total, ya que igual que el modelo P2P el sistema es elástico a usarse tanto con un crecimiento descontrolado de usuarios como en poblaciones muy pequeñas, pero además, tiene la optimización por parte del servidor que hace que cuando la población es demasiado pequeña modifique los tiempos para que las rondas se puedan llevar a cabo y cuando la población se ha desbordado también los modifica para evitar el "thundering herd problem" que ocurre cuando todos los dispositivos se encuentran compitiendo por participar en la ronda saturando al sistema que ha de evaluarlos.

c) Sé crítico y trata de presentar tu propia posición sobre las arquitecturas de aprendizaje federado. ¿Cuáles son los principales desafíos desde tu propia perspectiva?

Creo que las arquitecturas de aprendizaje federado son un paso adelante en cuanto a la inteligencia artificial aplicada al procesamiento de datos.

Ofrecen más seguridad y eliminan las desventajas temporales y espaciales de tener que acumular la cantidad de datos suficiente como para poder poner a un algoritmo a trabajar con ellos.

Por otro lado, dado el actual endurecimiento en cuanto a la política de privacidad de datos en la Unión Europea este tipo de arquitectura podría suponer una forma de poder ser competitivos en este sentido y hacer ciencia de datos en sectores tan sensibles como el de salud, accediendo a las conclusiones y las estadísticas de las enfermedades, tratamientos, diagnósticos, … sin leer ni tener posesión en ningún momento del historial médico de los pacientes.

Los problemas que presenta la arquitectura según el artículo me parecen fácilmente salvables a medida que el proyecto avance y se haga uso de algoritmos más eficientes.

En cuanto a los desafíos que presenta, bajo mi punto de vista, serian más de saber comunicar qué es lo que hace, cómo protege la privacidad y qué uso hace de los dispositivos. Como ya he comentado, creo que lo más complicado es la "fobia" al uso de nuestros datos, cuando precisamente veo este sistema como una muy buena solución para preservar nuestra intimidad.

Las compañías que lo implementen deberán hacer una inversión potente en seguridad y asegurarse (lo ideal sería que hubiese un cierto apoyo y garantía gubernamental, que se legislara qué medidas han de tomar las compañías para que el usuario final tenga la certeza y la garantía de que no se le está engañando) que en todo momento se están aplicando las mejores prácticas para garantizar la protección de los datos del usuario y para obtener datos lo más "limpios" y veraces posibles.

Con mi opinión tampoco quiero arremeter frontalmente contra la ley de protección de datos, me parece que la preservación es necesaria, pero a la vez veo el gran poder que estos pueden tener para mejorar la vida y la salud de las personas (los datos no solo sirven para vendernos cosas), por lo tanto, hallar la forma de poder estudiarlos y a la vez ser respetuoso con la intimidad de los individuos es, para mí, un avance de gran futuro y utilidad.

En conclusión, es un proyecto fascinante, que bien trabajado y respaldado puede dar un sinfín de posibilidades de mejora para nuestro día a día y nuestra salud entre otros campos de interés, es una tecnología por la que yo apostaría y aunque seguramente se mejore o supere en los años venideros representa una disrupción muy positiva en el campo del aprendizaje computacional.

(Coulouris, Dollimore, Kindberg, & Blair, Distributed Systems: Concepts and design, 2012)

(Bonawitz, y otros, 2019)

(Europa Press, 2019)

(Hao, 2019)

(Wikipedia, 2019)

REFERENCIAS

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., . . . Roselander, J. (22 de marzo de 2019). *Cornell University*. Obtenido de https://arxiv.org/pdf/1902.01046.pdf

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2012). 1.5 Challenges. En *Distributed Systems: Concepts and Design* (págs. 32-41). Essex: Pearson.

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2012). 1.5.1 Challenges: Heterogeneity. En *Distributed Systems: Concepts and Design* (pág. 33). Essex: Pearson.

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2012). 1.5.3 Challenges: Security. En *Distributed Systems: Concepts and Desing* (pág. 35). Essex: Pearson.

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2012). 1.5.5 Challenges: Failure handling. En *Distributed Systems: Concepts and Design* (págs. 37-38). Essex: Pearson.

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2012). 1.5.7 Challenges: Transparency. En *Distributed Systems: Concepts and Design* (págs. 39-41). Essex: Pearson.

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2012). 11.1.1 Introduction: Threats and attacks. En *Distributed Systems: Concepts and Desing* (págs. 483-484). Essex: Pearson.

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2012). 2.3.1 Architectural models: Architectural elements. En *Distributed Systems: Concepts and Design* (págs. 62-64). Essex: Pearson.

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2012). 2.3.1 Architectural models: Architectural elements. En *Distributed Systems: Concepts and Desing* (pág. 66). Essex: Pearson.

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2012). 2.4.2 Fundamental models: Failure model. En *Distributed Systems: Concepts and Design* (págs. 83-87). Essex: Pearson.

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2012). *Distributed Systems: Concepts and design.* Essex: Pearson.

Europa Press. (19 de abril de 2019). *ABC*. Obtenido de https://www.abc.es/tecnologia/informatica/software/abci-siguiente-paso-machine-learning-aprendizaje-colaborativo-desde-telefonos-moviles-201704121426_noticia.html?ref=https%3A%2F%2Fwww.google.com%2F

Hao, K. (26 de marzo de 2019). *MIT Technology Review*. Obtenido de https://www.technologyreview.es/s/11017/aprendizaje-federado-la-nueva-arma-de-ia-para-asegurar-la-privacidad

Wikipedia. (02 de agosto de 2019). Obtenido de https://en.wikipedia.org/wiki/Thundering_herd_problem