Big Data Aplicado

Computación distribuida

Contenido

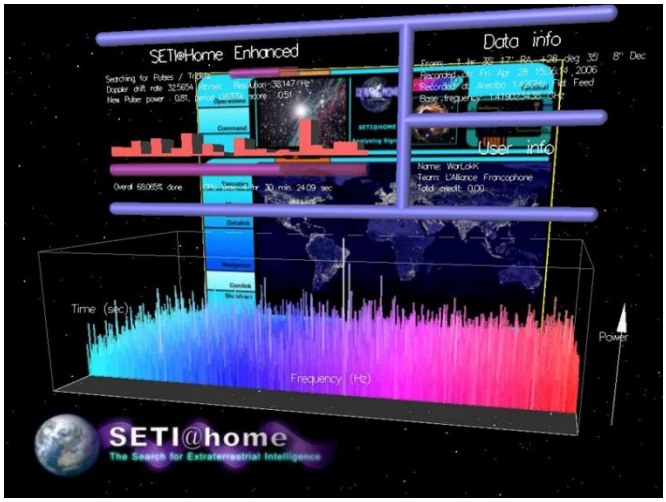
Computación distribuida

La computación distribuida es un modelo de resolución de problemas que utiliza un gran número de ordenadores, organizados en clústeres, que se comunican entre sí a través de una red.

En lugar de depender de una sola máquina, la computación distribuida divide las tareas entre varias máquinas, lo que permite manejar grandes volúmenes de datos y realizar cálculos intensivos de manera más eficiente.

Al estar físicamente separadas, estas máquinas no pueden utilizar una memoria compartida, así que transmiten mensajes y datos a través de una red.

Por ejemplo, **SETI@home** (https://setiathome.berkeley.edu/) fue un proyecto de computación distribuida que utilizaba computadoras personales conectadas a Internet para analizar señales de radio en busca de inteligencia extraterrestre. Los usuarios descargaban un software que procesaba pequeños fragmentos de datos y enviaba los resultados de vuelta a los servidores del proyecto (salvapantallas). Este enfoque permitía aprovechar la de procesamiento no utilizada de miles de computadoras voluntarias.



¿Cómo funciona la computación distribuida?

La computación distribuida funciona mediante computadoras que se pasan mensajes entre sí dentro de la arquitectura de los sistemas distribuidos. Los protocolos o reglas de comunicación crean una dependencia entre los componentes del sistema distribuido. Esta interdependencia se llama acoplamiento, y hay dos tipos principales.

Acoplamiento flexible

En el acoplamiento flexible, los componentes están débilmente conectados para que los cambios en uno de ellos no afecten al otro. Por ejemplo, las computadoras del cliente y del servidor pueden estar débilmente acopladas por tiempo. Los mensajes del cliente se agregan a una cola del servidor, y el cliente puede así continuar con sus otras funciones hasta que el servidor responda a su mensaje.

Acoplamiento ajustado

Los sistemas distribuidos de alto rendimiento suelen usar un acoplamiento ajustado. Las redes de área local rápidas suelen conectar varias computadoras, lo que crea un clúster. En la computación en clúster, cada computadora se configura para realizar la misma tarea. Los sistemas de control central, denominados middleware de agrupación, controlan y programan las tareas y coordinan la comunicación entre las distintas computadoras.

Tipos tradicionales de arquitectura de computación distribuida

En la computación distribuida, se diseñan aplicaciones que pueden ejecutarse en varias computadoras en lugar de en una sola. Esto se consigue al diseñar el software de forma que las distintas computadoras realicen diferentes funciones y se comuniquen para desarrollar la solución final. Siguiendo el principio de transparencia, se presentan como una unidad funcional, simplificando el funcionamiento al margen de la arquitectura utilizada.

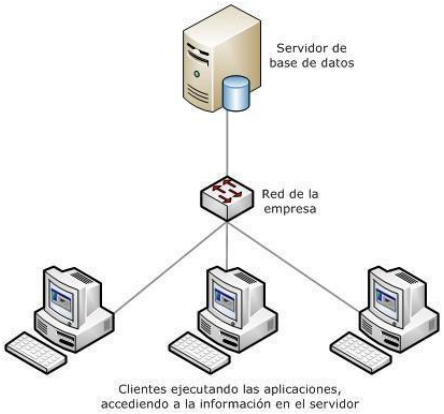
Existen cuatro tipos tradicionales de arquitectura distribuida.

Arquitectura cliente-servidor

Es la más común para la organización del software en un sistema distribuido.

- Clientes: actúan como instancia de entrada e interfaz de usuario, y realizan peticiones a los servidores.
- Servidores: sincronizan y administran el acceso a los recursos. Reciben solicitudes, realizan el procesamiento requerido y responden a las solicitudes de los clientes con datos, resultados o información de estado.

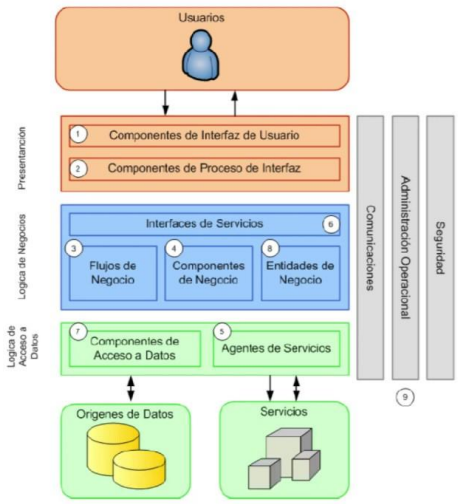
La limitación son los cuellos de botella que pueden provocar los servidores, especialmente cuando varios clientes realizan peticiones simultáneamente.



Arquitectura multinivel (modelo de capas)

- Capa de presentación: Interfaz de usuario.
- Capa de lógica de negocio: Procesamiento de datos y reglas de negocio.
- Capa de datos: Almacenamiento y administración de los datos. Responsables de la recuperación de los datos y de su coherencia.

Al dividir la responsabilidad de los servidores, los sistemas distribuidos de tres o más niveles reducen los cuellos de botella en las comunicaciones y mejoran el rendimiento de la computación distribuida.

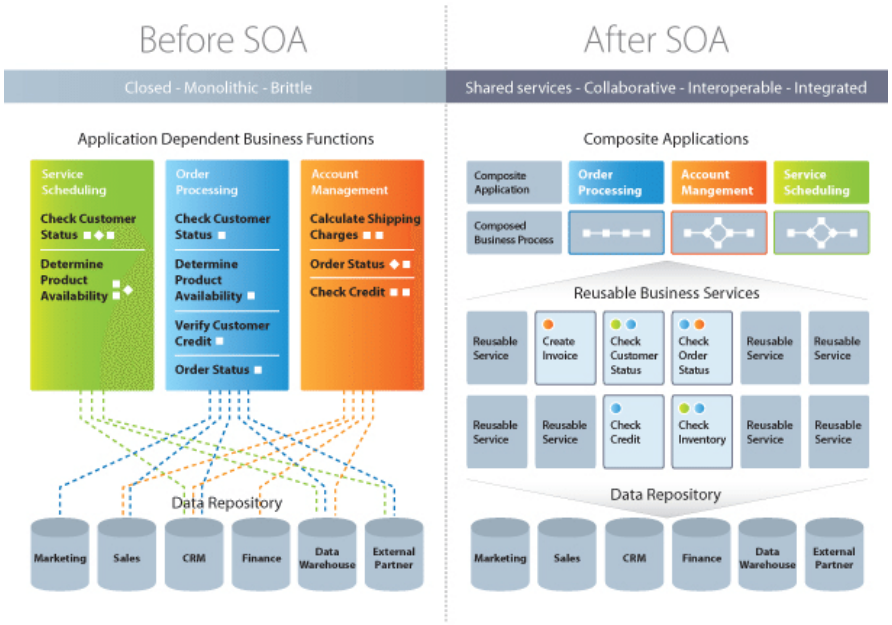


Arquitectura Orientada a Servicios (SOA)

Organiza las funcionalidades de una aplicación en servicios independientes que se comunican entre sí. Cada servicio realiza una función específica y se puede reutilizar en diferentes aplicaciones.

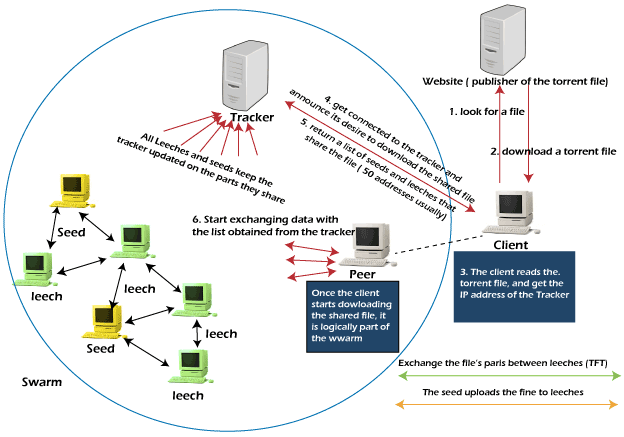
Los principios clave de SOA incluyen:

- **Interoperabilidad**: Los servicios pueden comunicarse a través de diferentes plataformas y lenguajes.
- **Reusabilidad**: Los servicios están diseñados para ser reutilizables en múltiples aplicaciones.
- **Modularidad**: Los servicios son componentes modulares que pueden ser desarrollados, desplegados y mantenidos de manera independiente.



Arquitectura entre pares (peer to peer)

Los sistemas distribuidos entre pares asignan responsabilidades iguales a todas las computadoras conectadas a la red. No hay separación entre las computadoras cliente y servidores, y cualquier computadora puede desempeñar todas las responsabilidades. La arquitectura entre pares se hace popular para el intercambio de contenidos, la transmisión de archivos y las redes blockchain.



Arquitectura distribuida MapReduce

Es un modelo de programación diseñado para procesar grandes volúmenes de datos en paralelo, utilizando un clúster de computadoras. Este modelo fue popularizado por Google y es una parte integral del ecosistema Hadoop.

Componentes Clave de MapReduce

JobTracker (Servidor Maestro):

- Función: Coordina la distribución de tareas y gestiona los recursos del clúster. Recibe las solicitudes de trabajo de los usuarios y las divide en tareas más pequeñas.
- Responsabilidades: Asigna tareas a los TaskTrackers, monitoriza su progreso y maneja la reejecución de tareas en caso de fallos.

TaskTrackers (Servidores Esclavos):

- Función: Ejecutan las tareas asignadas por el JobTracker. Cada nodo del clúster tiene un TaskTracker.
- Responsabilidades: Ejecutan las funciones de Map y Reduce, y reportan el estado de las tareas al JobTracker.

Fases del Proceso MapReduce

Fase Map:

- Descripción: Los datos de entrada se dividen en fragmentos y se distribuyen a los nodos del clúster. Cada nodo aplica la función Map a sus fragmentos, transformando los datos en pares clave-valor.
- Ejemplo: Si el objetivo es contar palabras en un conjunto de documentos, la función Map podría tomar cada documento y emitir pares (palabra, 1) para cada palabra encontrada.

Fase Shuffle and Sort:

- Descripción: Los pares clave-valor generados por la fase Map se redistribuyen y agrupan por clave. Esto asegura que todos los valores asociados con la misma clave se envíen al mismo nodo para la fase Reduce.
- Función: Ordena y agrupa los datos para prepararlos para la reducción.

Fase Reduce:

- Descripción: Los nodos aplican la función Reduce a los grupos de pares clave-valor. Esta función combina los valores asociados con la misma clave para producir un resultado final.

- Ejemplo: Continuando con el ejemplo de contar palabras, la función Reduce podría sumar todos los valores asociados con cada palabra para obtener el conteo total de cada palabra.

Beneficios de MapReduce

- **Escalabilidad**: Permite procesar grandes volúmenes de datos distribuyendo la carga de trabajo entre múltiples nodos.
- **Tolerancia a Fallos**: Si un nodo falla, las tareas pueden reasignarse a otros nodos, asegurando la continuidad del procesamiento.
- **Simplicidad de Programación**: Proporciona una abstracción sencilla para el procesamiento paralelo, permitiendo a los desarrolladores centrarse en la lógica de negocio sin preocuparse por la gestión del clúster.

Ejemplo de Uso

Hadoop: Utiliza HDFS (Hadoop Distributed File System) para almacenar datos distribuidos y MapReduce para procesarlos. Cada nodo en el clúster ejecuta tareas de Map y Reduce, coordinadas por el JobTracker y los TaskTrackers.

MapReduce es una herramienta poderosa para el procesamiento de grandes volúmenes de datos, especialmente en entornos donde la escalabilidad y la tolerancia a fallos son cruciales.

Computación paralela

La computación paralela es una forma de computación distribuida de acoplamiento ajustado. En el procesamiento en paralelo, todos los procesadores tienen acceso a la memoria compartida para intercambiar información entre ellos. En cambio, en el procesamiento distribuido, cada procesador tiene memoria privada (memoria distribuida). Los procesadores usan el paso de mensajes para intercambiar información.

Computación en malla

La computación en malla es una computación distribuida a gran escala que hace hincapié en el rendimiento y la coordinación entre varias redes. Internamente, cada malla actúa como un sistema de computación de acoplamiento ajustado. Sin embargo, externamente, las mallas están acopladas de forma más suelta. Cada red de mallas realiza funciones individuales y comunica los resultados a otras mallas.

Nuevas arquitecturas de computación distribuida

1. Arquitectura de Microservicios

- Divide una aplicación en un conjunto de servicios pequeños e independientes que se comunican entre sí a través de APIs. Cada microservicio se encarga de una funcionalidad específica y puede ser desarrollado, desplegado y escalado de manera independiente.
- Ventajas: Facilita la escalabilidad, mejora la resiliencia y permite una mayor flexibilidad en el desarrollo y despliegue de aplicaciones.
- Por ejemplo, en una aplicación de comercio electrónico:
  - **Capa de presentación**: Una interfaz web que se comunica con los microservicios a través de REST APIs.
  - **Capa de lógica de negocio**: Microservicios para gestión de productos, procesamiento de pagos y gestión de usuarios.
  - **Capa de datos**: Microservicios que interactúan con bases de datos para almacenar y recuperar información.

2. Serverless Computing (Computación sin Servidor)

- Permite a los desarrolladores ejecutar código sin tener que gestionar servidores. Los proveedores de servicios en la nube ejecutan el código en respuesta a eventos y gestionan automáticamente la infraestructura necesaria.
- Ventajas: Reduce los costos operativos, mejora la escalabilidad y permite a los desarrolladores centrarse en el código en lugar de la infraestructura.

3. Edge Computing (Computación en el Borde)

- Procesa los datos cerca de la fuente de generación (por ejemplo, dispositivos IoT) en lugar de enviarlos a un centro de datos centralizado. Esto reduce la latencia y el ancho de banda necesario.
- Ventajas: Mejora la velocidad de procesamiento, reduce la latencia y permite una mejor gestión de los datos en tiempo real.

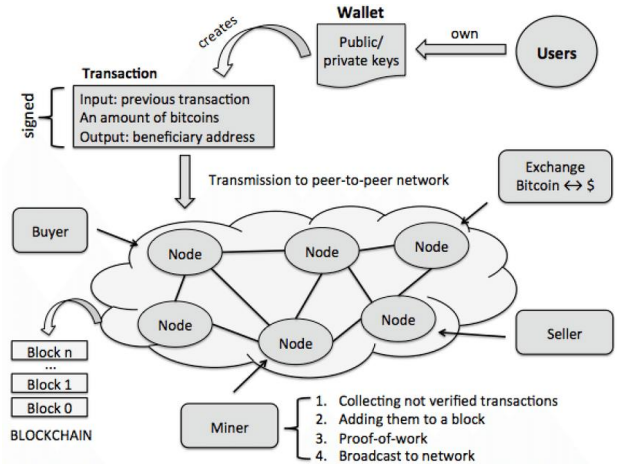
4. Arquitectura de Contenedores y Orquestación (Kubernetes)

- Utiliza contenedores para empaquetar aplicaciones y sus dependencias, asegurando que se ejecuten de manera consistente en diferentes entornos. Kubernetes es una plataforma de orquestación que gestiona el despliegue, escalado y operación de contenedores.

- Ventajas: Facilita la portabilidad, mejora la eficiencia de recursos y simplifica la gestión de aplicaciones distribuidas.

5. Arquitectura de Blockchain

- Utiliza una red distribuida de nodos para mantener un registro inmutable y seguro de transacciones. Cada bloque de datos está vinculado al anterior, formando una cadena.
- Ventajas: Proporciona seguridad, transparencia y descentralización, lo que es ideal para aplicaciones que requieren confianza y verificación sin intermediarios.



6. Arquitectura de Computación Cuántica

- Utiliza principios de la mecánica cuántica para realizar cálculos que serían inviables para las computadoras clásicas. Los qubits pueden representar y procesar más información que los bits tradicionales.
- Ventajas: Promete resolver problemas complejos en áreas como criptografía, optimización y simulación molecular mucho más rápido que las computadoras clásicas.

Estas arquitecturas están transformando la manera en que diseñamos y gestionamos sistemas distribuidos, ofreciendo nuevas oportunidades y desafíos en el campo de la computación.

Ventajas de la computación distribuida

Escalabilidad horizontal

La capacidad de agregar más nodos al clúster para aumentar la potencia de procesamiento. Esto contrasta con la escalabilidad vertical, que se basa en mejorar los recursos de una sola máquina.

Disponibilidad / Tolerancia a fallos

Si un nodo del clúster falla, el sistema puede continuar funcionando con los nodos restantes. La replicación de datos en múltiples nodos es crucial para esta capacidad.

Consistencia

Las computadoras de un sistema distribuido comparten información y duplican datos entre ellos, pero el sistema administra automáticamente la coherencia de datos en todas las computadoras. De este modo, se obtiene el beneficio de la tolerancia a los fallos sin comprometer la coherencia de datos.

Transparencia

Los sistemas de computación distribuida proporcionan una separación lógica entre el usuario y los dispositivos físicos. Puede interactuar con el sistema como si se tratara de una única computadora sin preocuparse de la instalación y configuración de las máquinas individuales. Puede tener diferentes hardware, middleware, software y sistemas operativos que trabajan juntos para que su sistema funcione sin problemas.

Eficiencia

Los sistemas distribuidos ofrecen un rendimiento más rápido con un uso óptimo de los recursos del hardware subyacente. Como resultado, puede administrar cualquier carga de trabajo sin preocuparse por el fallo del sistema debido a los picos de volumen o por la infrautilización del hardware.

Además, se pueden utilizar ordenadores de bajo coste para construir el clúster, en lugar de supercomputadoras especializadas.

Deslocalización

El procesamiento se puede distribuir a los nodos donde se encuentran los datos, lo que reduce la necesidad de transferir grandes cantidades de datos a través de la red.

Inconvenientes de la computación distribuida

Trabajar con computación distribuida ofrece muchas ventajas, pero también presenta varios inconvenientes que es importante tener en cuenta:

Complejidad

- Gestión y Mantenimiento: La configuración y mantenimiento de sistemas distribuidos pueden ser complicados debido a la necesidad de coordinar múltiples componentes con diferentes versiones de software y hardware.

- Desarrollo: El desarrollo de aplicaciones para sistemas distribuidos requiere habilidades especializadas y una comprensión profunda de la concurrencia y la sincronización.

Seguridad

- Vulnerabilidades: Los sistemas distribuidos son más susceptibles a ataques debido a la mayor superficie de ataque. Cada nodo puede ser un punto de entrada potencial para atacantes.
- Protección de Datos: Asegurar la integridad y confidencialidad de los datos en tránsito y en reposo es más complejo en un entorno distribuido.

Sincronización y Coordinación

- Sincronización de Datos: Mantener la consistencia de los datos entre nodos puede ser un desafío, especialmente en sistemas con alta latencia o nodos geográficamente dispersos.
- Coordinación de Tareas: Asegurar que las tareas se completen en el orden correcto y manejar las dependencias entre tareas puede ser complicado.

Tolerancia a Fallos

- Detección de Fallos: Identificar y manejar fallos en un sistema distribuido es más difícil que en un sistema centralizado. Los fallos pueden ser intermitentes y difíciles de reproducir.
- Recuperación: Implementar mecanismos de recuperación y redundancia para asegurar la continuidad del servicio puede ser costoso y complejo.

Costos

- Infraestructura: La configuración inicial y el mantenimiento de un sistema distribuido requieren una inversión significativa en hardware y software.
- Operación: Los costos operativos pueden ser altos debido a la necesidad de monitorear y gestionar múltiples nodos y redes.

Enrutamiento y Latencia

- Enrutamiento: La comunicación entre nodos en diferentes ubicaciones puede complicar el enrutamiento y aumentar la latencia.
- Latencia: La latencia de red puede afectar el rendimiento de las aplicaciones distribuidas, especialmente aquellas que requieren respuestas rápidas.

Interoperabilidad

- Estándares y Protocolos: Asegurar que todos los componentes del sistema puedan comunicarse y trabajar juntos de manera efectiva requiere el uso de estándares y protocolos bien definidos.