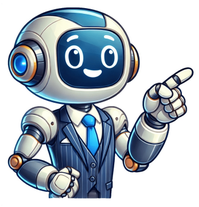
Click to prove you're human



Un sistema distribuido es un conjunto de programas informáticos que utilizan recursos computacionales en varios nodos de cálculo distintos para lograr un objetivo compartido común. Este tipo de sistemas, también denominados "computación distribuida" o "bases de datos distribuidas", usan nodos distintos para comunicarse y sincronizarse a través de una red común. Estos nodos suelen representar dispositivos de hardware físicos diferentes, pero también pueden representar procesos de software diferentes u otros sistemas encapsulados recursivos. La finalidad de los sistemas distribuidos es eliminar los cuellos de botella o los puntos de error centrales de un sistema. Los sistemas de computación distribuida tienen las características siguientes: Recursos compartidos: los sistemas distribuidos pueden compartir hardware, software o datos Procesamiento simultáneo: varias máquinas pueden procesar la misma función a la vez Escalabilidad: la capacidad computacional y de procesamiento puede escalarse hacia arriba según sea necesario cuando se añaden máquinas adicionales Detección de errores: los errores se pueden detectar más fácilmente Transparencia: un nodo puede acceder a otros nodos del sistema y comunicarse con ellos Un sistema de computación centralizado es aquel en el que todos los cálculos los realiza un solo ordenador en una ubicación. La principal diferencia entre un sistema centralizado y un sistema distribuido es el patrón de comunicación entre los nodos del sistema. El estado de un sistema centralizado se encuentra dentro de un nodo central al que los clientes acceden a través de un método personalizado. Todos los nodos de un sistema centralizado acceden al nodo central, lo que puede sobrecargar y ralentizar la red. Los sistemas centralizados tienen un punto de error único. Este no es el caso de los sistemas distribuidos. Una arquitectura de microservicios es un tipo de sistema distribuido porque descompone una aplicación en distintos componentes o "servicios". Por ejemplo, una arquitectura de microservicios puede tener servicios que correspondan a funciones empresariales (pagos, usuarios, productos, etc.), en los que cada uno de los componentes correspondientes gestione la lógica empresarial para dicha responsabilidad. Luego, el sistema tendrá varias copias redundantes de los servicios de modo que no exista ningún punto de error central para un servicio. El seguimiento distribuido es un método que se utiliza para perfilar o supervisar el resultado de una solicitud que se ejecuta en un sistema distribuido. La supervisión de un sistema distribuido puede ser complicada porque cada nodo tiene su propio flujo de registros y métricas. Para obtener una visión precisa de un sistema distribuido, estas distintas métricas de nodos deben agruparse en una vista holística. Normalmente, las solicitudes que se envían a los sistemas distribuidos no acceden a todo el conjunto de nodos del sistema, sino a un conjunto parcial o a una ruta a través de los nodos. El seguimiento distribuido señala las rutas a las que se accede habitualmente a través de un sistema distribuido y permite que los equipos las analicen y las supervisen. El seguimiento distribuido se instala en cada nodo del sistema y permite que los equipos pidan al sistema información sobre el estado de los nodos y el rendimiento de la solicitud. Los sistemas distribuidos suelen contribuir a mejorar la fiabilidad y el rendimiento del sistema. La fiabilidad se mejora eliminando los puntos de error centrales y los cuellos de botella. Los nodos de un sistema distribuido ofrecen redundancia, de modo que, si un nodo falla, hay otros que pueden sustituirlos y reparar el error. El rendimiento se mejora porque los nodos pueden escalarse fácilmente en sentido horizontal y vertical. Si un sistema se somete a una carga extensiva, pueden añadirse nodos adicionales para ayudar a absorber dicha carga. También es posible aumentar la capacidad de un nodo concreto para gestionar una carga extensiva. Sin embargo, uno de los inconvenientes que presentan estos sistemas puede ser la "expansión del desarrollo", que implica que los sistemas acaban siendo excesivamente complejos y difíciles de mantener. Esta mayor complejidad puede complicar a los equipos las tareas de organización, gestión y mejora de estos sistemas. Parte de la dificultad puede ser entender la relación que hay entre los distintos componentes o quién posee un componente de software concreto. Esto hace que sea difícil saber cómo modificar los componentes para maximizar su rendimiento y evitar que ello tenga un impacto negativo no solo en los componentes dependientes, sino también en los clientes. Cuando un sistema distribuido tiene varios repositorios, puede que sea necesario utilizar herramientas especializadas como Atlassian Compass para gestionar y organizar el código del sistema. Hay muchos tipos de sistemas distribuidos. Estos son los más habituales: Una arquitectura cliente-servidor se divide en dos responsabilidades principales. El cliente se encarga de la presentación de la interfaz de usuario, que luego se conecta al servidor a través de la red. El servidor se encarga de gestionar la lógica empresarial y el estado. Una arquitectura cliente-servidor puede degradarse fácilmente a una arquitectura centralizada si el servidor no es redundante. Una configuración cliente-servidor perfectamente distribuida tendrá varios nodos de servidor para distribuir las conexiones de los clientes. Las arquitecturas cliente-servidor más modernas son clientes que se conectan a un sistema de distribución encapsulado en el servidor. Una arquitectura por capas amplía la arquitectura cliente-servidor. En ella, el servidor se descompone en otros nodos granulares, que desacoplan responsabilidades adicionales del servidor backend, como el procesamiento y la gestión de los datos. Estos nodos adicionales se usan para procesar de forma asíncrona las tareas de larga duración y liberar al resto de los nodos del backend para que puedan centrarse en responder a las solicitudes del cliente e interactuar con el almacén de datos. En un sistema distribuido punto a punto, cada nodo contiene la instancia completa de una aplicación. No hay separación de nodos entre la presentación y el procesamiento de datos. Un nodo contiene la capa de presentación y las capas de gestión de datos. Los otros nodos pueden contener todos los datos de estado del sistema. Los sistemas punto a punto tienen la ventaja de ser extremadamente redundantes. Cuando un nodo punto a punto se inicializa y se lleva online, descubre otros nodos, se conecta a ellos y sincroniza su estado local con el estado de todo el sistema. Por ello, si en un sistema punto a punto un nodo falla, los demás nodos no se verán afectados. Además, el sistema punto a punto persistirá. La arquitectura orientada a servicios (SOA) es anterior a los microservicios. La principal diferencia entre la SOA y los microservicios es el alcance de los nodos: el alcance de los nodos de los microservicios está a nivel de la función. En los microservicios, un nodo encapsula la lógica empresarial para gestionar un conjunto específico de funciones, como el procesamiento de los pagos. Los microservicios contienen varios nodos de lógica empresarial dispares que interactúan con nodos de bases de datos independientes. Comparativamente, los nodos SOA encapsulan toda una aplicación o toda una división empresarial. El límite de servicio de los nodos SOA suele ser un sistema de base de datos completo dentro del nodo. Los microservicios son una alternativa más popular a la arquitectura SOA debido a las ventajas que ofrecen. Son más modulares, ya que permiten a los equipos reutilizar funcionalidades que proporcionan los nodos de servicio pequeños. También son más sólidos y permiten realizar un escalado vertical y horizontal más dinámico. Muchas aplicaciones modernas utilizan sistemas distribuidos. Las aplicaciones web y móviles con mucho tráfico son sistemas distribuidos. Los usuarios se conectan siguiendo un modelo cliente-servidor, donde el cliente es un navegador web o una aplicación móvil. El servidor es pues su propio sistema distribuido. Los servidores web modernos siguen un patrón de sistema por capas. Se utiliza un equilibrador de carga para delegar las solicitudes a muchos nodos lógicos de servidor que se comunican a través de sistemas de cola de mensajes. Kubernetes es una herramienta muy utilizada en los sistemas distribuidos porque puede crear un sistema distribuido a partir de una colección de contenedores. Los contenedores crean nodos del sistema distribuido y Kubernetes orquesta la comunicación de red entre esos nodos, además de gestionar el escalado dinámico horizontal y vertical de los nodos en el sistema. Otro buen ejemplo de sistema distribuido son las criptomonedas, como Bitcoin y Ethereum, que son sistemas distribuidos punto a punto. Cada nodo de una red de criptomoneda es una réplica independiente de todo el historial del libro mayor de la moneda. Cuando un nodo de moneda se lleva online, se pone en funcionamiento conectándose a otros nodos y descargando su copia completa del libro mayor. Además, las criptomonedas tienen clientes o "carteras" que se conectan a los nodos del libro mayor a través del protocolo JSON RPC. Los sistemas distribuidos gozan de una gran aceptación y se usan en la mayoría de las experiencias de software modernas. Las aplicaciones de redes sociales, los servicios de streaming de vídeo y los sitios de comercio electrónico son solo algunas de las tecnologías que utilizan sistemas distribuidos. Los sistemas centralizados evolucionan de forma natural hacia sistemas distribuidos para gestionar el escalado. El uso de microservicios es una práctica habitual y muy extendida para crear sistemas distribuidos. Aunque los sistemas distribuidos son más difíciles de crear y mantener, Atlassian Compass elimina esta complejidad. Esta plataforma para desarrolladores permite desplazarse fácilmente por una arquitectura distribuida, ya que reúne en un lugar centralizado que admite búsquedas la información desconectada sobre los procesos de ingeniería y los equipos que colaboran en ellos. 0 calificaciones0% encontró este documento útil (0 votos)326 vistasEste documento resume los conceptos fundamentales de los sistemas distribuidos basados en la web. Describe los lenguajes HTML y XML usados para crear documentos web, el protocolo HTTP que pe…Título y descripción mejorados con IAGuardarGuardar SISTEMAS DISTRIBUIDOS BASADOS EN WEB para más tarde0%0% encontró este documento útil, undefined Sistemas Distribuidos basados en la web El origen y razón de las páginas web interactivasArquitecturas Cliente-servidor • Normalmente para consultar, modificar y administrar los datos de una organización • El programa cliente era la interfaz inteligente entre el usuario y la base de datos Servidor de BD clienteInconvenientes de las arquitecturas de 2 capas • Aplicaciones monolíticas difíciles de mantener • Toda la inteligencia está en el cliente • Los servidores son sólo servidores de datos • Distribución del código que cambia • Poca reusabilidad del código • No está orientado al desarrollo de componentes • No se puede usar de cualquier lado • Poca seguridad • Mucho tráfico de datosArquitecturas basadas en la web • El cliente es el browser • La aplicación misma está en el servidor web • Puede existir un servidor de aplicaciones Servidor de aplicaciones Servidor web cliente Servidor de BDVentajas del esquema de n-capas • Clara separación de las funciones de control de la interfaz y presentación de datos con la lógica de la aplicación • Reusabilidad de componentes • Independencia de la interfaz del cliente y la arquitectura de datos • Mejores posibilidades de balancear la carga • Uso de protocolos abiertos ¿Qué es la World-wide Web ? • WWW es un sistema de hipermedia interactivo desarrollado sobre Internet. • Hipermedia: La idea de hipermedia es la de juntar texto, imágenes, audio y vídeo dentro de un mismo envoltorio llamado documento • Además, incluye referencias a otros documentos del mismo tipo ¿Cómo nace la World-wide Web ? • El primer precedente del WWW se puede encontrar en un tratado escrito por Vannevar Bush titulado ``As We May Think'', de 1954, en el que básicamente animaba a los científicos a hacer más accesible a todo el mundo sus conocimientos y experiencia. En este tratado surgieron las nociones de hipertexto e hipermedia • Desde entonces, muchos han sido los sistemas hipermedia que se han creado. Como ejemplo cabe destacar el Augment de Doug Englebart en los 70, y en los 90 el WWW de Tim Berners-Lee, TBL, creado en el Laboratorio Europeo de Partículas Físicas (CERN) en Ginebra, Suiza. TBL se basó en muchas fuentes para crear el WWW: Pero la web necesita un protocolo para intercambiar documentos de hipermedia Internet Para ello hay una conversación entre el que pide un documento Y el que lo “sirve” (invisible para el usuario del browser) Esta conversación se realiza según el protocolo HTTP Arquitectura Web Para abrir una página Web en un navegador, normalmente se teclea el correspondiente URL o se “clickea” sobre en el “link” oportuno. Una vez que se solicita esta petición mediante el protocolo HTTP y la recibe el servidor Web, éste localiza la página Web en su sistema de directorios y la envía de vuelta al navegador que la solicitó,¿ Qué necesito para “levantar” mi sitio web ? El Servidor Web El servidor Web es un programa que corre sobre el servidor que escucha las peticiones HTTP que le llegan y las satisface. Dependiendo del tipo de la petición, el servidor Web buscará una página Web o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML al cliente o navegador que realizó la petición. El servidor Web va a ser fundamental en el desarrollo de las aplicaciones del lado del servidor, server side applications, que vayamos a construir, ya que se ejecutarán en él. ¿De dónde lo saco, cómo lo instalo ? • Hay varios gratis que se pueden bajar de la Internet • JBOSS, APACHE TOMCAT • Para instalarlo basta echar a correr el archivo bajado (doble click) • Obviamente, el computador debe estar en la internet y tener una dirección IP fija y un nombre de dominioJ2EE • Es un conjunto de especificaciones que implementan una arquitectura abierta de n-capas sobre la web • Incluye muchos elementos que se habían desarrollado en forma “independiente” anteriormente • Interoperabilidad gracias a XML y SOAPLos principales elementos de la arquitectura J2EE • Cliente • Puede ser un programa “stand alone”, un applet dentro de un browser o el browser mismo que contacta un sevlet, un jsp o un web service • Contenedor web • Contiene (y sabe como hacerlos correr) las páginas html, los servlets, JSP y web services • Contenedor de aplicaciones • Contiene y administra a los EJB • Servidor de base de datosQué va dónde Portlets y Web services Servidor de aplicaciones Servidor web Cliente: Browser web Servidor de BD Servidor web y de aplicaciones Páginas HTML -Java Script -JSP Enterprise Java Beans Comunicación por medio de JDBCJ2EE icluye • Acceso a bases de datos (JDBC) • Utilización de directorios distribuidos (JNDI) • Acceso a métodos remotos (RMI, CORBA, SOAP) • Funciones de correo electrónicoEl protocolo Http • Hypertext transfer protocol • Especifica la forma como el cliente pide las cosas al servidor y cómo el servidor las manda al cliente • Define un lenguaje de comunicación con reglas sintácticas y gramaticalesInteraccion browser-servidor • Cuando se ingresa una URL al navegador web pidiendo un archivo el navegador manda la siguiente información: GET /index.jsp HTTP 1.1 Accept: image/gif, image/jpeg, application/vnd.msexcel, application/vnd.ms-powerpoint, application/msword, Accept-Language: de Accept-Encoding: gzip, deflate User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1) Host: localhost:8000 Connection: Keep-Alive