ÁREA 4. SOLUCIONES DE CÓMPUTO INTELIGENTE

SUBÁREA 4.3 CÓMPUTO DISTRIBUIDO

TEMAS IMPORTANTE:

**Sistemas Operativos (Stallings):**

**3.1 ¿Qué es un proceso?**

En términos generales, un proceso es un programa en ejecución en una computadora. Es la entidad fundamental sobre la cual el sistema operativo realiza la planificación y asignación de recursos de la computadora, como el tiempo de CPU, la memoria y los dispositivos de entrada/salida.

Más específicamente, un proceso puede definirse como un programa en ejecución que consta de dos partes principales: el código del programa (también conocido como texto del programa) y los datos asociados con ese programa (como las variables, los registros de estado y el espacio de memoria asignado). Cada proceso tiene su propio espacio de memoria y recursos asignados por el sistema operativo, lo que le permite funcionar de manera aislada de otros procesos en el sistema.

Además, cada proceso tiene un identificador único llamado PID (Identificador de Proceso), que el sistema operativo utiliza para realizar un seguimiento del estado del proceso y para proporcionar servicios como la comunicación entre procesos y la sincronización.

**4.1 Procesos e hilos**

Un proceso es un programa en ejecución en la computadora y consta de dos partes principales: el código del programa y los datos asociados con ese programa. Cada proceso tiene su propio espacio de memoria y recursos asignados por el sistema operativo, lo que le permite funcionar de manera aislada de otros procesos en el sistema. Además, cada proceso tiene un identificador único llamado PID (Identificador de Proceso), que el sistema operativo utiliza para realizar un seguimiento del estado del proceso y para proporcionar servicios como la comunicación entre procesos y la sincronización.

Por otro lado, un hilo (también conocido como subproceso) es una entidad dentro de un proceso que puede ejecutar una parte del código del programa de manera independiente. Los hilos comparten el mismo espacio de memoria y recursos del proceso principal, lo que les permite compartir información y recursos más fácilmente que los procesos. Sin embargo, cada hilo tiene su propio estado de ejecución, lo que significa que puede ejecutar diferentes secciones de código al mismo tiempo.

La principal ventaja de usar hilos es que pueden mejorar la eficiencia del sistema, ya que los hilos pueden ejecutarse en paralelo y aprovechar los múltiples núcleos de procesamiento de la CPU. Además, los hilos también pueden ser útiles para la programación multitarea, ya que varios hilos dentro de un proceso pueden ejecutar diferentes tareas simultáneamente.

En resumen, mientras que un proceso es un programa en ejecución que consta de código y datos asociados, los hilos son entidades dentro de un proceso que pueden ejecutar diferentes secciones de código de manera independiente y compartiendo recursos con otros hilos dentro del mismo proceso.

**14.2 Paso de mensajes distribuido**

El paso de mensajes distribuido es una técnica de programación utilizada en sistemas distribuidos para permitir la comunicación entre procesos que se ejecutan en diferentes computadoras. En lugar de compartir memoria compartida, los procesos se comunican intercambiando mensajes a través de la red.

En un sistema distribuido, cada proceso tiene su propio espacio de direcciones de memoria y no puede acceder directamente a la memoria de otro proceso. Por lo tanto, el paso de mensajes es una forma de permitir que los procesos se comuniquen y coordinen sus acciones.

Para enviar un mensaje en un sistema distribuido, un proceso emisor codifica el mensaje en un formato aceptado por el sistema de paso de mensajes y lo envía a un proceso receptor identificado por su dirección. El proceso receptor recibe el mensaje, lo decodifica y realiza la acción apropiada en respuesta al mensaje.

El paso de mensajes distribuido ofrece varias ventajas, como la independencia de la arquitectura de hardware y la ubicación física de los procesos. Además, este enfoque puede ser utilizado para permitir la comunicación entre procesos en diferentes sistemas operativos y lenguajes de programación.

Sin embargo, también presenta algunos desafíos, como la necesidad de implementar mecanismos de seguridad y confiabilidad para garantizar que los mensajes sean entregados de manera confiable y que la comunicación entre procesos no sea interceptada o manipulada por terceros.

**14.3 Llamadas a procedimiento remote**

Las llamadas a procedimientos remotos (RPC, por sus siglas en inglés) son un mecanismo utilizado en sistemas distribuidos para permitir que un proceso en una computadora realice una llamada a un procedimiento en otra computadora y reciba el resultado de esa llamada. RPC es una técnica de comunicación entre procesos que permite a los procesos distribuidos trabajar juntos como si estuvieran en la misma computadora.

Cuando un proceso cliente realiza una llamada a un procedimiento remoto, esta llamada se empaqueta en un mensaje que se envía a través de la red a un proceso servidor que implementa el procedimiento. El servidor procesa la solicitud y envía una respuesta al cliente. El cliente espera hasta que recibe la respuesta del servidor antes de continuar.

RPC permite que los procesos distribuidos interactúen de manera transparente, como si estuvieran en la misma computadora, lo que facilita la programación de aplicaciones distribuidas. Sin embargo, también presenta desafíos, como la necesidad de garantizar la seguridad y la integridad de los mensajes enviados a través de la red, así como la necesidad de gestionar la disponibilidad y escalabilidad del servidor.

En resumen, las llamadas a procedimientos remotos son una técnica de comunicación entre procesos que permite que los procesos distribuidos trabajen juntos como si estuvieran en la misma computadora. Es una forma común de implementar servicios distribuidos en sistemas distribuidos y puede ser utilizada en una variedad de aplicaciones, incluyendo sistemas de bases de datos distribuidos, sistemas de archivos distribuidos y sistemas de computación en la nube.

**14.4 Clusters**

Un cluster es un conjunto de computadoras o servidores interconectados que trabajan juntos para realizar una tarea común. Los clusters son utilizados en sistemas de alta disponibilidad, computación distribuida y procesamiento de grandes volúmenes de datos.

En un cluster, cada computadora se conoce como un nodo. Los nodos trabajan juntos para proporcionar servicios de alta disponibilidad y escalabilidad, y para ejecutar aplicaciones de procesamiento intensivo de datos. Los nodos pueden estar conectados mediante una red de área local o una red de área amplia.

Los clusters se utilizan en una variedad de aplicaciones, como el procesamiento de datos masivos, la simulación y el modelado científico, la inteligencia artificial y el aprendizaje automático, y la informática en la nube.

Los clusters son especialmente útiles para mejorar el rendimiento y la disponibilidad de sistemas críticos. Por ejemplo, un sistema de base de datos distribuido puede utilizar un cluster para proporcionar redundancia y alta disponibilidad. Si uno de los nodos falla, los otros nodos pueden tomar el control del servicio y garantizar que los datos estén disponibles para los usuarios.

Además, los clusters también pueden proporcionar una escalabilidad horizontal, lo que permite a los sistemas distribuidos agregar más nodos para procesar más datos y soportar más usuarios. Esto permite a los sistemas distribuidos adaptarse a las demandas cambiantes de los usuarios y proporcionar un mejor rendimiento y escalabilidad en general.

**14.7 Clusters Beowulf y Linux**

Un cluster Beowulf es un tipo de cluster de computadoras de alta rendimiento (HPC) construido con hardware de bajo costo y software de código abierto. El nombre Beowulf proviene del poema épico inglés del mismo nombre que cuenta la historia del héroe Beowulf, que se asocia con la idea de un guerrero solitario luchando contra enemigos poderosos. En el mundo de la informática, esto se aplica a la idea de un cluster de computadoras que pueden competir en poder de procesamiento con sistemas más grandes y costosos.

Los clusters Beowulf utilizan software de código abierto como el sistema operativo Linux y aplicaciones HPC de software libre como OpenMPI, MPICH y LAM-MPI. Estos componentes de software permiten que los nodos del cluster se comuniquen entre sí y trabajen juntos para completar tareas de procesamiento intensivo.

En un cluster Beowulf, un nodo de computadora actúa como un maestro o servidor y se encarga de la coordinación de las tareas en el cluster. Los nodos del cluster se comunican entre sí a través de una red de alta velocidad, como InfiniBand, para compartir datos y recursos.

Linux es un sistema operativo ampliamente utilizado en clusters Beowulf debido a su capacidad de personalización y capacidad de trabajar con hardware de bajo costo. Linux también es de código abierto, lo que permite a los usuarios personalizar y modificar el sistema operativo según sus necesidades.

En resumen, un cluster Beowulf es un cluster de computadoras de alta velocidad construido con hardware de bajo costo y software de código abierto, especialmente Linux. Este tipo de cluster es utilizado en aplicaciones de procesamiento intensivo de datos y en el campo de la informática de alta velocidad para proporcionar un rendimiento y una escalabilidad mejorados a un costo más bajo que los sistemas de hardware propietarios.

**Introduction to Parallel Computing:**

**1.2 How—There Are Three Prevailing Types of Parallelism**

En el contexto de la computación paralela, el autor Peter Pacheco, en su libro "An Introduction to Parallel Programming", menciona que existen tres tipos principales de paralelismo: el paralelismo de datos, el paralelismo de tareas y el paralelismo de instrucciones.

El paralelismo de datos se refiere a la capacidad de procesar múltiples elementos de datos al mismo tiempo. En este tipo de paralelismo, los datos se dividen en pequeñas partes y se procesan simultáneamente en diferentes núcleos de procesamiento o unidades de procesamiento gráfico (GPU). Este tipo de paralelismo es comúnmente utilizado en aplicaciones de procesamiento de imágenes, video, audio y simulación numérica.

El paralelismo de tareas se refiere a la capacidad de procesar múltiples tareas al mismo tiempo. En este tipo de paralelismo, las tareas se dividen en pequeñas partes y se ejecutan simultáneamente en diferentes núcleos de procesamiento. Este tipo de paralelismo es comúnmente utilizado en aplicaciones de redes de alta velocidad, procesamiento de datos en tiempo real y computación de alta velocidad.

El paralelismo de instrucciones se refiere a la capacidad de procesar múltiples instrucciones de programa al mismo tiempo. En este tipo de paralelismo, las instrucciones de programa se dividen en pequeñas partes y se ejecutan simultáneamente en diferentes núcleos de procesamiento. Este tipo de paralelismo es comúnmente utilizado en aplicaciones de procesamiento de señales digitales, criptografía y procesamiento de bases de datos.

En resumen, el paralelismo es una técnica que permite procesar múltiples tareas o datos al mismo tiempo para mejorar el rendimiento y la eficiencia de los sistemas computacionales. Los tres tipos principales de paralelismo son el paralelismo de datos, el paralelismo de tareas y el paralelismo de instrucciones, cada uno con diferentes aplicaciones y técnicas de implementación.

**2.4 The Impact of Communication**

En el contexto de la programación paralela, la comunicación entre procesos es un factor crítico que puede tener un gran impacto en el rendimiento y la escalabilidad de las aplicaciones. La comunicación entre procesos se refiere al intercambio de datos entre diferentes procesos que se ejecutan en diferentes núcleos de procesamiento o en diferentes máquinas en un sistema distribuido.

La comunicación puede ser necesaria para coordinar las tareas de diferentes procesos, sincronizar el estado de diferentes procesos, compartir datos y recursos, y enviar mensajes y señales entre diferentes procesos. Sin embargo, la comunicación también puede ser costosa en términos de tiempo y recursos, y puede limitar el rendimiento y la escalabilidad de las aplicaciones si no se gestiona adecuadamente.

El impacto de la comunicación en el rendimiento y la escalabilidad de las aplicaciones depende de varios factores, como el tamaño y la frecuencia de los mensajes, la latencia y el ancho de banda de la red de comunicación, el número de procesos involucrados y el algoritmo de comunicación utilizado. Para minimizar el impacto de la comunicación en el rendimiento de las aplicaciones paralelas, es necesario optimizar el diseño y la implementación de la comunicación y elegir el algoritmo de comunicación adecuado según las necesidades de la aplicación y las características del sistema.

En resumen, la comunicación entre procesos es un factor crítico en la programación paralela que puede tener un gran impacto en el rendimiento y la escalabilidad de las aplicaciones. Es importante gestionar adecuadamente la comunicación para minimizar su impacto en el rendimiento y elegir el algoritmo de comunicación adecuado según las necesidades de la aplicación y las características del sistema.

**3.1 SharedMemory ProgrammingModel**

El modelo de programación de memoria compartida es un paradigma de programación paralela en el que varios procesos o hilos de ejecución acceden a un espacio de memoria común para compartir datos y comunicarse entre sí. En este modelo, los procesos pueden acceder directamente a la memoria compartida sin la necesidad de comunicarse explícitamente a través de mensajes o señales.

En general, el modelo de programación de memoria compartida se divide en dos categorías principales: memoria compartida con bloqueo y memoria compartida sin bloqueo.

La memoria compartida con bloqueo utiliza técnicas de exclusión mutua para garantizar que solo un proceso pueda acceder a un recurso compartido en un momento dado. Esto se logra mediante el uso de bloqueos o semáforos para sincronizar el acceso a los datos compartidos. Aunque la memoria compartida con bloqueo es fácil de implementar y entender, puede resultar en problemas de escalabilidad y rendimiento cuando se utilizan muchos procesos.

La memoria compartida sin bloqueo, por otro lado, utiliza técnicas de sincronización menos restrictivas para permitir que varios procesos accedan simultáneamente a los datos compartidos. Estas técnicas incluyen el uso de primitivas de sincronización de hardware como el intercambio atómico y la carga-almacenamiento atómico. La memoria compartida sin bloqueo puede proporcionar un mejor rendimiento y escalabilidad que la memoria compartida con bloqueo, pero es más difícil de implementar y depurar.

En resumen, el modelo de programación de memoria compartida es una forma común de programación paralela en la que varios procesos comparten un espacio de memoria común para comunicarse y compartir datos. La memoria compartida con bloqueo utiliza técnicas de exclusión mutua para sincronizar el acceso a los datos compartidos, mientras que la memoria compartida sin bloqueo utiliza técnicas de sincronización menos restrictivas para permitir que varios procesos accedan simultáneamente a los datos compartidos.

**4.1 Distributed Memory Computers Can Execute in Parallel**

Las computadoras con memoria distribuida son sistemas paralelos en los que múltiples procesadores o nodos de procesamiento están conectados a través de una red de comunicación. Cada procesador tiene su propia memoria local y no puede acceder directamente a la memoria de otros procesadores. En cambio, los procesadores se comunican a través de la red de comunicación para intercambiar datos y coordinar sus tareas.

En los sistemas con memoria distribuida, la programación paralela se basa en el modelo de paso de mensajes, en el que los procesos se comunican enviando y recibiendo mensajes a través de la red de comunicación. Cada proceso tiene una identidad única y una dirección en la red que se utiliza para enviar y recibir mensajes. Los mensajes pueden contener datos o comandos, y se utilizan para coordinar las tareas de los procesos y compartir datos entre ellos.

La programación paralela en sistemas con memoria distribuida puede ser más complicada que en sistemas con memoria compartida, debido a la necesidad de gestionar la comunicación explícitamente y distribuir los datos entre los procesadores. Sin embargo, los sistemas con memoria distribuida pueden escalar a mayores niveles de paralelismo que los sistemas con memoria compartida, ya que pueden agregar más procesadores simplemente agregando más nodos a la red.

En resumen, los sistemas con memoria distribuida son sistemas paralelos en los que múltiples procesadores están conectados a través de una red de comunicación y utilizan el modelo de paso de mensajes para coordinar sus tareas y compartir datos. Aunque la programación paralela en sistemas con memoria distribuida puede ser más complicada que en sistemas con memoria compartida, los sistemas con memoria distribuida pueden escalar a mayores niveles de paralelismo y pueden ser más adecuados para aplicaciones de alta escalabilidad y distribuidas.

**Sistemas Operativos Distribuidos (Tanenbaum):**

* 1. **¿Qué es un sistema distribuido?**

Un sistema distribuido es un conjunto de componentes de hardware y software interconectados que trabajan juntos para realizar una tarea común. Estos componentes se encuentran físicamente separados y se comunican a través de una red de comunicación, en lugar de compartir una memoria centralizada. Los componentes pueden ser computadoras, servidores, dispositivos móviles, sensores, etc.

En un sistema distribuido, cada componente se ejecuta en su propio entorno de ejecución independiente, lo que permite que los componentes se agreguen y eliminen fácilmente del sistema sin afectar el funcionamiento general. Cada componente del sistema distribuido tiene su propia capacidad de procesamiento y memoria, lo que permite que el sistema se escale horizontalmente agregando más componentes.

La comunicación entre los componentes de un sistema distribuido se realiza a través de la red, utilizando protocolos y servicios de comunicación específicos. Los componentes pueden enviar y recibir mensajes para compartir información y coordinar sus acciones, lo que permite que el sistema distribuido funcione como un todo coherente.

Los sistemas distribuidos se utilizan en una amplia variedad de aplicaciones, desde sistemas de telecomunicaciones hasta sistemas de comercio electrónico y aplicaciones en la nube. La ventaja principal de los sistemas distribuidos es su capacidad para escalar horizontalmente, lo que les permite manejar cargas de trabajo cada vez mayores al agregar más componentes al sistema.

**1.5 Aspectos del diseño**

El diseño de sistemas distribuidos es un proceso complejo que debe tener en cuenta diversos aspectos para garantizar su funcionalidad, escalabilidad, eficiencia y fiabilidad. A continuación, se describen algunos de los aspectos más importantes del diseño de sistemas distribuidos:

* Arquitectura: la arquitectura de un sistema distribuido define la estructura y organización de los componentes, protocolos y servicios utilizados para la comunicación entre ellos. Existen diferentes arquitecturas, como la arquitectura cliente-servidor, la arquitectura peer-to-peer y la arquitectura orientada a servicios, cada una con sus ventajas y desventajas.
* Comunicación: la comunicación entre los componentes de un sistema distribuido es esencial para su correcto funcionamiento. Es necesario definir los protocolos de comunicación, el formato de los mensajes y los mecanismos de detección y corrección de errores.
* Tolerancia a fallos: los sistemas distribuidos deben ser capaces de continuar operando incluso en presencia de fallos de hardware o software. Es necesario diseñar mecanismos de tolerancia a fallos, como la replicación de datos, la detección de fallos y la recuperación ante fallos.
* Seguridad: los sistemas distribuidos pueden estar expuestos a diversos riesgos de seguridad, como ataques de hackers, malware o robo de datos. Es necesario diseñar medidas de seguridad, como la autenticación, el cifrado de datos y la auditoría de accesos.
* Escalabilidad: los sistemas distribuidos deben ser capaces de escalar horizontalmente para manejar cargas de trabajo cada vez mayores. Es necesario diseñar la arquitectura, la comunicación y los mecanismos de tolerancia a fallos de forma que se puedan agregar componentes sin afectar la funcionalidad general del sistema.
* Gestión de recursos: los sistemas distribuidos suelen utilizar recursos compartidos, como el ancho de banda de la red o la capacidad de almacenamiento. Es necesario diseñar mecanismos para gestionar el acceso a estos recursos y evitar conflictos entre los componentes.
* Interoperabilidad: los sistemas distribuidos pueden estar compuestos por componentes de diferentes proveedores y tecnologías. Es necesario diseñar los protocolos y servicios de comunicación de forma que permitan la interoperabilidad entre los diferentes componentes.

En resumen, el diseño de sistemas distribuidos debe tener en cuenta diversos aspectos, como la arquitectura, la comunicación, la tolerancia a fallos, la seguridad, la escalabilidad, la gestión de recursos y la interoperabilidad, para garantizar su funcionalidad y eficiencia en entornos distribuidos y escalables.

**2.4 Llamada a un procedimiento Remoto (RPC)**

La Llamada a un Procedimiento Remoto (RPC, por sus siglas en inglés) es un modelo de programación que permite a un programa llamar a un procedimiento o función que se ejecuta en un sistema remoto, como si estuviera llamando a una función local. El RPC es utilizado en sistemas distribuidos para permitir que los procesos que se ejecutan en diferentes sistemas se comuniquen entre sí y cooperen para realizar una tarea común.

El funcionamiento de un RPC se basa en la idea de que el cliente que solicita la ejecución de un procedimiento remoto no tiene que preocuparse por cómo se realiza la llamada al procedimiento en el servidor remoto, ni por cómo se transmiten los datos entre el cliente y el servidor. Todo esto es manejado por el sistema RPC.

En un RPC, el cliente envía una solicitud al servidor indicando el nombre del procedimiento a ejecutar y los parámetros necesarios. El servidor ejecuta el procedimiento y devuelve el resultado al cliente. Para que esto sea posible, el sistema RPC proporciona una interfaz de programación que abstrae la complejidad de la comunicación entre los componentes del sistema distribuido.

El uso de un RPC presenta varias ventajas. En primer lugar, permite a los programadores trabajar con una interfaz de programación más simple y abstracta, en lugar de tener que preocuparse por la complejidad de la comunicación entre componentes. Además, facilita la construcción de sistemas distribuidos, ya que los componentes pueden estar en diferentes sistemas y pueden comunicarse entre sí de manera transparente.

Sin embargo, el uso de un RPC también presenta algunos desafíos. Por ejemplo, es necesario asegurarse de que la interfaz de programación sea coherente en todos los sistemas, lo que puede ser un desafío en entornos heterogéneos. Además, se deben tener en cuenta las cuestiones de seguridad, como la autenticación y la encriptación, para evitar ataques malintencionados y proteger la privacidad de los datos transmitidos.

**4.1 Hilos**

Los hilos (también conocidos como threads) son unidades de ejecución de un programa que comparten el mismo espacio de memoria y recursos del sistema. Los hilos permiten que un programa realice múltiples tareas de forma concurrente, lo que puede mejorar significativamente la eficiencia y el rendimiento del sistema.

En un programa con múltiples hilos, cada hilo tiene su propio contador de programa y pila de llamadas, pero comparte el mismo espacio de memoria y recursos del sistema con los demás hilos. Esto permite que los hilos compartan datos y recursos, lo que puede simplificar la programación y reducir el consumo de recursos.

Existen dos tipos principales de hilos: los hilos de usuario y los hilos de kernel. Los hilos de usuario son implementados en el espacio de usuario del sistema operativo y son manejados por la biblioteca de hilos de la aplicación. Los hilos de kernel son creados y manejados directamente por el sistema operativo y tienen mayor control sobre los recursos del sistema, aunque también son más costosos en términos de recursos.

Los hilos pueden ser utilizados para dividir una tarea en múltiples sub-tareas que pueden ejecutarse en paralelo, lo que puede mejorar significativamente el rendimiento del sistema. También pueden ser utilizados para mejorar la capacidad de respuesta de una aplicación, ya que permiten que una tarea larga se divida en varias tareas más pequeñas y se ejecuten de forma concurrente, lo que reduce el tiempo de respuesta percibido por el usuario.

Sin embargo, el uso de hilos también presenta algunos desafíos. Por ejemplo, es necesario garantizar la sincronización y la comunicación adecuada entre los hilos para evitar problemas como la condición de carrera y la inconsistencia de datos. Además, el uso excesivo de hilos puede tener un efecto contrario y disminuir el rendimiento del sistema debido a la sobrecarga del sistema operativo y la competencia por los recursos del sistema.

**5.1 Diseño de los sistemas distribuidos de archivos**

El diseño de sistemas distribuidos de archivos (SDAF) es un desafío importante en la construcción de sistemas distribuidos. Los SDAF permiten que múltiples usuarios compartan y accedan a archivos de manera transparente, como si estuvieran almacenados en un único sistema de archivos.

El diseño de un SDAF debe abordar varias cuestiones importantes, como la gestión de los metadatos del archivo, la distribución de los datos y la resolución de conflictos. Entre los aspectos a considerar en el diseño de SDAF se encuentran los siguientes:

* Consistencia: es importante garantizar que los usuarios vean el mismo estado de los archivos, independientemente de dónde estén almacenados. Para lograr esto, se utilizan técnicas de replicación y sincronización de datos.
* Escalabilidad: el sistema debe ser capaz de manejar grandes cantidades de datos y usuarios. La escalabilidad se puede lograr mediante la distribución de datos y la paralelización de las operaciones
* Tolerancia a fallos: dado que los sistemas distribuidos están sujetos a fallos, es importante diseñar el SDAF para que sea capaz de recuperarse de los fallos sin perder datos o interrumpir el acceso a los archivos. Esto se puede lograr mediante la replicación de datos y la redundancia en el hardware y el software.
* Seguridad: es importante garantizar la seguridad y la privacidad de los datos almacenados en el sistema distribuido. Se deben implementar medidas de seguridad, como la autenticación y el cifrado de datos, para proteger los archivos de accesos no autorizados.
* Rendimiento: el rendimiento del SDAF es crítico para una experiencia de usuario satisfactoria. Es importante considerar la latencia de la red, el ancho de banda y la distribución de datos para optimizar el rendimiento del sistema.
* Interoperabilidad: los SDAF deben ser compatibles con diferentes sistemas operativos y plataformas de hardware para permitir una amplia adopción y uso.

En resumen, el diseño de un SDAF debe tener en cuenta la consistencia, escalabilidad, tolerancia a fallos, seguridad, rendimiento e interoperabilidad para proporcionar una experiencia de usuario satisfactoria y confiable.

**Sistemas Operativos Modernos (Tanenbaum):**

**2.1 Proceso:**

En los sistemas operativos modernos, un proceso es una instancia de un programa en ejecución que tiene su espacio de memoria privado y recursos asignados, incluyendo recursos de CPU, memoria, archivos abiertos, etc. Un proceso puede contener uno o varios hilos de ejecución, y puede comunicarse con otros procesos a través de mecanismos de comunicación interproceso (IPC) proporcionados por el sistema operativo.

**2.2 Hilos:**

Un hilo es una unidad básica de ejecución que pertenece a un proceso. Un proceso puede tener varios hilos en ejecución simultáneamente, y cada hilo puede ejecutar su propia secuencia de instrucciones. Los hilos dentro de un proceso comparten recursos como la memoria y los archivos abiertos, lo que permite una mayor eficiencia en la utilización de los recursos del sistema.

**8.4.7 Grids (Mallas):**

Una grid, o malla en español, es un tipo de sistema distribuido diseñado para compartir recursos informáticos a través de una red de computadoras geográficamente distribuidas. Las grids permiten la cooperación entre organizaciones y usuarios independientes para resolver problemas complejos que requieren grandes cantidades de recursos informáticos. Un ejemplo de aplicación de una grid es la búsqueda de fármacos y el procesamiento de imágenes médicas. Las grids se basan en software de middleware que gestiona la distribución de tareas y recursos, y proporciona mecanismos para la comunicación y la seguridad en la red. Además, la tecnología de virtualización se utiliza a menudo para permitir una mayor flexibilidad en la asignación de recursos.

**Introduction to Grid Computing IBM:**

**Capítulo 1. Qué es la Computación en Malla:**

La Computación en Malla es un tipo de computación distribuida que implica compartir recursos informáticos a través de una red, generalmente la Internet. La Computación en Malla permite a las organizaciones y a las personas acceder y utilizar grandes cantidades de potencia de procesamiento, almacenamiento de datos y otros recursos, sin necesidad de poseer o administrar la infraestructura de hardware y software subyacente. La Computación en Malla normalmente involucra un gran número de computadoras o servidores dispersos geográficamente que están conectados por una red de alta velocidad y pueden trabajar juntos en tareas de cómputo a gran escala.

**Capítulo 2. Beneficios de la Computación en Malla:**

La Computación en Malla ofrece una serie de beneficios para las organizaciones y las personas, incluyendo:

* Mayor potencia de procesamiento: La Computación en Malla permite a los usuarios aprovechar la potencia de procesamiento colectiva de muchos sistemas interconectados, proporcionando un nivel mucho mayor de potencia de procesamiento que el que sería posible con un solo sistema.
* Ahorro de costos: Al compartir recursos informáticos entre múltiples organizaciones, la Computación en Malla puede ayudar a reducir los costos totales de TI, ya que elimina la necesidad de que las organizaciones individuales inviertan en y mantengan su propia infraestructura informática.
* Mejora de la escalabilidad: La Computación en Malla facilita la escalabilidad de los recursos informáticos según sea necesario, sin necesidad de adquirir nuevo hardware o software
* Mejora de la eficiencia: La Computación en Malla permite a los usuarios utilizar de manera más eficiente los recursos informáticos, distribuyendo las cargas de trabajo entre múltiples sistemas y evitando la necesidad de ejecutar tareas secuencialmente en un solo sistema.
* Mejora de la colaboración: La Computación en Malla puede facilitar la colaboración entre individuos y organizaciones, permitiéndoles compartir datos, recursos y experiencia a través de una red.
* Incremento de la innovación: Al facilitar el acceso y el uso de los recursos informáticos, la Computación en Malla puede permitir que las personas y las organizaciones emprendan nuevas iniciativas de investigación y desarrollo, y exploren ideas innovadoras que de otra manera no serían factibles.

**Cloud Computing:**

Cloud Computing es un modelo de entrega de servicios de computación en el que los recursos informáticos, como servidores, almacenamiento, aplicaciones y servicios, se entregan a través de Internet y se escalan dinámicamente según la demanda. En lugar de adquirir y mantener infraestructura y software costosos, los usuarios pueden pagar por el uso de los recursos en función de la cantidad y la duración de su uso.

El Cloud Computing se divide en tres categorías principales de servicios:

* Software como Servicio (SaaS): En este modelo, el proveedor de Cloud Computing proporciona una aplicación completa que se ejecuta en su infraestructura y se entrega a los usuarios finales a través de Internet. Ejemplos de servicios SaaS incluyen aplicaciones de correo electrónico basadas en la nube como Gmail y servicios de almacenamiento en la nube como Dropbox.
* Plataforma como Servicio (PaaS): En este modelo, el proveedor de Cloud Computing proporciona una plataforma completa que permite a los desarrolladores crear, ejecutar y administrar aplicaciones en línea. La plataforma incluye un sistema operativo, herramientas de desarrollo, servicios de base de datos y aplicaciones de servidor web. Ejemplos de servicios PaaS incluyen Microsoft Azure y Google App Engine.
* Infraestructura como Servicio (IaaS): En este modelo, el proveedor de Cloud Computing proporciona recursos de infraestructura virtualizados, como servidores, almacenamiento y redes, que los usuarios pueden utilizar para construir y ejecutar sus propias aplicaciones y servicios. Los usuarios tienen control completo sobre la configuración y el mantenimiento de los recursos virtuales. Ejemplos de servicios IaaS incluyen Amazon Web Services y Microsoft Azure.

El Cloud Computing ofrece una serie de beneficios, como la escalabilidad elástica, la reducción de costos, la disponibilidad global y la flexibilidad. Sin embargo, también plantea desafíos en términos de seguridad, privacidad, cumplimiento normativo y dependencia de los proveedores de servicios de Cloud Computing.