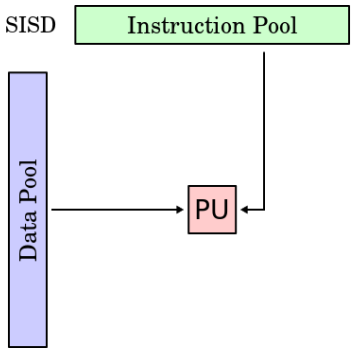
Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencias de la Computación
Sistemas Distribuidos
Arquitecturas de Sistemas Distribuidos

1 Taxonomía de Flynn

Michael J. Flynn propuso 4 modelos para generalizar estructuras de procesamiento según la cantidad de flujos de control y de datos:

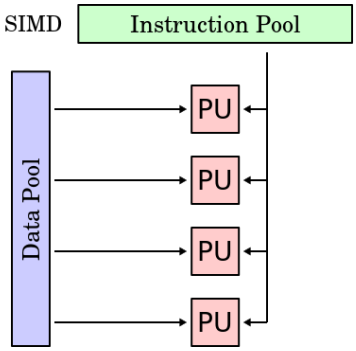
1.1 SISD: Single Instruction, Single Data

Un flujo de instrucciones que opera sobre un flujo de datos. Es lo que se conoce como una arquitectura Von Neumann. Hay un procesamiento secuencial de las instrucciones. Un solo flujo de instrucciones que opera sobre datos que están en un mismo componente de memoria.



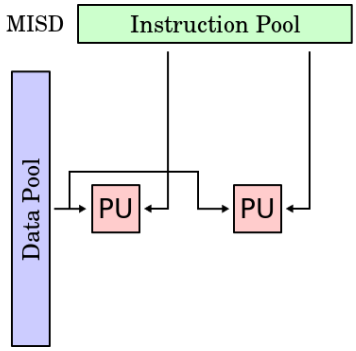
1.2 SIMD: Single Instruction, Multiple Data

Un flujo de instrucciones que opera sobre distintos flujos de datos. Lo que ocurre es que cada instrucción del flujo se ejecuta sobre distintos flujos de datos, se reparten. Esta arquitectura explota el paralelismo de datos y se utiliza mucho en aplicaciones matriciales.



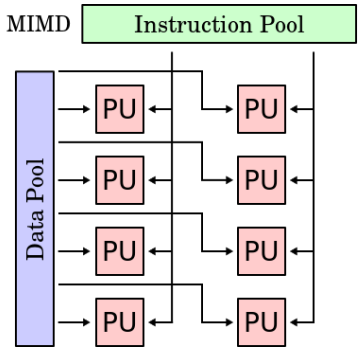
1.3 MISD: Multiple Instruction, Single Data

Hay múltiples flujos de instrucciones sobre un solo flujo de datos. Esta arquitectura no es muy común.



1.4 MIMD: Multiple Instruction, Multiple Data

Múltiples flujos de instrucciones sobre múltiples flujos de datos. Es el caso más general de paralelismo. Hay hartos procesadores que pueden ejecutar distintas instrucciones sobre distintos datos sin necesitar estar sincronizados.



2 Tipos de Paralelismo

Paralelismo de Control
en inglés se conoce como **Task Parallelism**, y como dice el nombre, se trata de una paralelización de tareas, donde más de una tarea se ejecuta al mismo tiempo que otra con los mismos datos.

Paralelismo de Datos
en este caso, se ejecuta la misma tarea pero con distintos componentes de datos.

3 Organización de Memoria

Se puede catalogar en dos tipos.

3.1 Memoria Distribuida

Cada procesador tiene una memoria propia y local. Por lo tanto, los accesos a memoria deben ser bien explícitos ya que esto en general se soluciona por paso de mensajes. Hay distintas topologías utilizadas para conectar los procesadores, entre ellas están: Bus, Ring, Star, Extended Star, Hierarchical, Mesh y Hypercube.

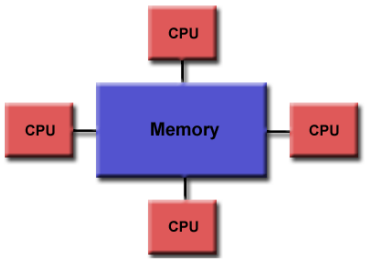
3.2 Memoria Compartida

en este caso todos los procesadores están conectados a una misma memoria compartida. La lectura y escritura de los procesadores se puede hacer al mismo tiempo. El problema es que se pueden generar cuellos de botella cuando muchos procesos desean acceder a memoria pero el uso de caches puede aliviar este problema.
La coherencia es otro tema que se debe cuidar. Por ejemplo, si dos procesos intentan modificar un dato al mismo tiempo sería problemático. Hay técnicas

como los locks y semáforos que lidian con este problema, materia vista en Sistemas Operativos.
Veremos a continuación dos arquitecturas de memoria compartida utilizadas para organizarla.

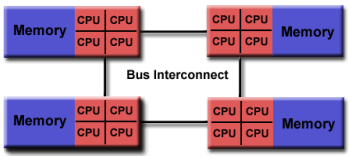
3.2.1 UMA: Uniform Memory Access

Todos los procesos comparten de forma uniforme la memoria. El tiempo de acceso a una sección de memoria es independiente del procesador que realiza el request. A continuación se muestra un diagrama de esta estructura:



3.2.2 NUMA: Non-uniform Memory Access

Hay distintas formas para acceder a la memoria. En general se aprovecha la localización de la memoria respecto al procesador que accede a ésta. Abajo se muestra un diagrama de esta arquitectura.



4 Volviendo a Sistemas Operativos...

Hay dos conceptos muy importantes mencionados en el primer apunte y vistos en Sistemas Operativos que será importante recordar:

Threads
Hilo de ejecución que corresponde a la división más pequeña de una instrucción que puede ser manejada por un scheduler de forma independiente de otros procesos o componentes. Técnicamente es un proceso pequeño y atómico (la palabra átomo proviene del griego y significa indivisible). Dependiendo del Sistema Operativo, dentro de un proceso puede haber varios threads que corran de forma concurrente.

Procesos

Ejecución de una o más instrucciones. Por ejemplo, la ejecución de la instancia un programa es un proceso. Un procesador no puede correr de forma "real" más de un proceso a la vez, esto se mencionó en el primer apunte. Solo se puede realizar una concurrencia "ilusoria" donde dos procesos se van turnando tan rápidamente que parece paralelismo. Con más de un procesador si se puede generar una concurrencia o paralelismo real.

5 Conceptos Importantes de Redes

es importante recordar los siguientes conceptos de redes, vistos en Sistemas Operativos.

5.1 Puerto de Red

Componente lógica o interfaz identificada por un número para comunicarse con otro componente de una red (puede ser incluso consigo mismo).

5.2 Socket de Red

representación lógica de una conexión que ocupa un puerto específico y un protocolo específico. Esta conexión se utiliza para enviar y recibir datos. Es la instanciación de una conexión en un nodo de la red.

5.3 Protocolos de Red

un Protocolo de Red define reglas y convenciones que se aplican en distintos aspectos de la comunicación entre componentes de una red. Estas convenciones afectan la forma en la que los componentes se comunican entre ellos e identifican. También tienen instrucciones específicas de como empaquetar y desempaquetar datos para crear un mensaje, los métodos que se deben aplicar en la presencia de un error y la sincronización de la comunicación. Hay muchos protocolos distintos, entre ellos están:

5.3.1 TCP: Transmission Control Protocol

Está en la capa de transporte. Protocolo de Internet basado en conexiones. El nodo envía un paquete a otro nodo y espera recibir una respuesta antes de seguir adelante. En otras palabras, el emisor se asegura de que el receptor haya recibido el paquete lo que hace que este protocolo tenga una alta confiabilidad a costa de sacrificar rapidez. Por ejemplo, los juegos online multiplayer.

5.3.2 UDP: User Datagram Protocol

Está en la capa de transporte. También es un protocolo de comunicación de Internet. El nodo envía paquetes sin esperar y asegurarse que el nodo receptor haya recibido correctamente los mensajes. Si algún paquete UDP se perdió y

no fue recibido, simplemente se perdió y no es enviado nuevamente. En este caso se obtiene una mayor rapidez sacrificando confiabilidad. Por ejemplo, un streaming.

5.3.3 HTTP: Hypertext Transfer Protocol

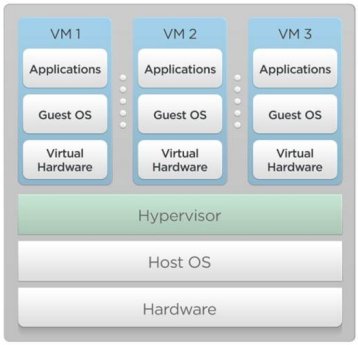
Está en la capa de aplicación. Utiliza TCP. Este protocolo permite la transferencia de información a través del World Wide Web. Establece la sintáxis y semántica de la comunicación. Este protocolo está orientado a transacciones y sigue un esquema petición y request entre un cliente y un servidor.

5.3.4 DNS: Domain Name System

Sistema jerárquico y descentralizado para nombrar servicios y sistemas conectados a Internet o a una red local. Traduce nombres de dominio (como el URL) a direcciones IP. El siguiente video es muy bueno para comprenderlo de forma rápida: https://www.youtube.com/watch?v=72snZctFFtA .

6 Virtualización

Tecnología que permite la creación de muchos ambientes aislados simulados a partir de un solo hardware real. Un Software llamado **Hypervisor** se conecta directamente con el Hardware real y en él se montan estos ambientes virtuales. Estos ambientes se llaman máquinas virtuales y dependen de la capacidad del Hypervisor para emular a la máquina real. Se virtualiza hardware y sistema operativo (llamados Guest OS). Es probable que ya hayan utilizado máquinas virtuales para correr un sistema operativo distinto al de su computador por necesidad de trabajar en otro ambiente. En Mac OS uno muy popular es Parallels o Virtual Box que corre en Windows, Mac OS, Linux y Solaris.
A continuación se muestra un diagrama que representa a este sistema:



Hay distintos tipos de virtualización:

6.1 Full Virtualization

en este tipo de virtualización, el Guest OS no tiene conocimiento de pertenecer a una máquina virtual, por lo tanto, el Host OS virtualiza hardware a través del cual el Guest OS cree que estará realmente ejecutando comandos. Todo es virtualizado, es una simulación completa del hardware de la máquina.

6.2 Paravirtualization

en este caso, el Guest OS tiene el conocimiento de ser un Guest. Los Guest OS son modificados de tal forma que tienen acceso a una interfaz similar al hardware real a través de la cual los comandos se dirigen directamente al hardware real. En **Full Virtualization** se virtualzaba el hardware real y los comandos se dirigían a ese hardware virtual. La intención de la Paravirtualización es reducir el tiempo de ejecución de operaciones del sistema virtual ya que es bastante más difícil correr comandos y operaciones en un ambiente virtualizado que en uno no virtualizado. Al no virtualizar al hardware realmente, la Paravirtualización es más liviana.

6.3 Hardware Assisted Virtualization

esta técnica de virtualización utiliza recursos del hardware del Host OS para realizar una **Full Virtualization** eficiente.

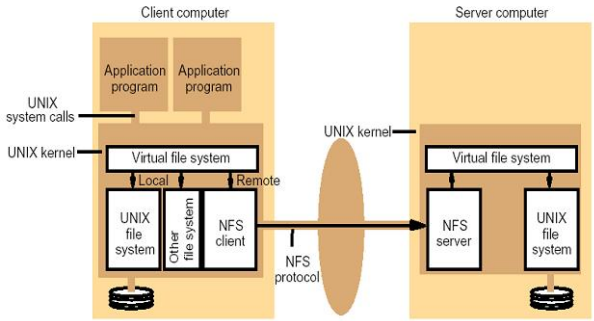
Existe un tipo de virtualización conocida como **Contenedores** como **Docker**. Los contenedores a diferencia de las Máquinas Virtuales no virtualizan Hardware por lo que son mucho más livianas. También comparten el mismo Guest OS lo que disminuye la aislación y por lo tanto, se reduce la seguridad ya que si se contamina un contenedor hay mayores posibilidades de que el resto de los contenedores se infecten. Dependiendo de lo que se desee hacer, los contenedores pueden ser mucho más útiles y están siendo utilizados por muchas empresas de servicios computacionales muy importantes. Google tiene su propio sistema de contenedores llamado **Kubernetes**. Kubernetes provee una mayor posibilidad de customización que Docker pero requiere un esfuerzo mucho mayor para instalarlo y configurarlo.

7 Sistemas de Archivos Distribuidos

Permiten compartir información en forma de archivos a través de una red utilizando almacenamiento. Si está bien diseñado, debiese funcionar igual de bien que un sistema de almcenamiento archivos local.
Permite a los usuarion acceder a sus archivos desde cualquier computador de la red. Existen distintos tipos o protocolos de Sistemas de Archivos Distribuidos.

7.1 NFS: Network File System

Protocolo de Sistema de Archivos desarrollado por **Sun Microsystems** (1984). Es centralizado, hay un **Servidor NFS**. Este protocolo permite montar todo o una parte de un sistema de archivos en este servidor. Esta aplicación sigue un modelo de cliente/servidor y permite a los clientes acceder a los archivos del servidor NFS como si fuesen locales. Se utilizan llamadas RPC entre clientes y servidores. Se puede utilizar en todos los sistemas Unix. Muchos problemas se solucionaron con la versión 4 como el tema de la seguridad, ahora hay un sistema basado en **Kerberos** (protocolo de autentificación de red).



7.2 AFS: Andrew File System

Utiliza más de un servidor manteniendo la transparencia, homogeneidad y apariencia de localidad que se espera de un sistema de archivos distribuido. Fue desarrollado en Carnegie Mellon. Implicó una mejora sobre los sistemas de archivos distribuidos del momento como una mejor escabilidad y seguridad. Utiliza **Kerberos** e implementa listas de control de acceso en los directorios para los usuarios.
Cada cliente tiene su propio cache en su sistema de archivos local, esto se hizo con la finalidad de aumentar la eficiencia luego de muchos requests de un mismo archivo. La versión 4 de NFS se vio influenciada por AFS.

7.3 Lustre File System

Sistema de archivos distribuidos usualmente utilizado en clusters. Se lanzó el año 2003 y tiene un kernel Linux. Se monta en múltiples servidores a diferencia de NFS. Debido a su gran escalabilidad y eficiencia, es utilizado ampliamente en super computadores que están en el ranking del **TOP500** (los top 500 super computadores más poderosos del mundo).
Está compuesto por 3 grandes unidades:

7.3.1 1 o más MDS (metadata servers)

Estos nodos almacenan nombres de archivos, de directorios, permisos de acceso y el layout de archivos. La metadata se almacena en un sistema de archivos local.

7.3.2 1 o más OSS: Object Storage Server

Almacenan datos de archivos en uno o más dispositivos **OST** (Object Storage Data Devices). La capacidad del sistema de archivos Lustre dependerá de la cantidad de estos dispositivos y de los clientes que puedan utilizarlos.

7.3.3 Clientes

Usuarios que utilicen el sistema. Se permite lectura y escritura concurrente y coherente.

7.4 OrangeFS

Sistema de archivos distribuido Open Source. Como Lustre, es utilizado en grandes clusters por compañías, universidades, empresas y laboratorios de investigación. Fue lanzado en el año 2011 lo que lo hace bastante nuevo. Este sistema de archivos tiene múltiples servidores y es descentralizado. A partir de la versión 4.6 forma parte del kernel de Linux. Se recomienda visitar la página del proyecto: http://www.orangefs.org/ .

7.5 HDFS: Hadoop Distributed File System

Sistema de archivos distribuido portable y escalable. Orientado fuertemente a clusters. Está escrito en Java para el framework de Hadoop (es un subproducto de Apache Hadoop). Los computadores conectados al cluster son conocidos como nodos. Se accede y almacenan los bloques de datos como si fuese un sistema de archivo homogéneo utilizando el modelo **MapReduce**.
La escritura se puede realizar solo por un nodo a la vez mientras que la lectura puede realizarse por uno o más a la vez. De esta forma, se simplifica la mantención de la coherencia.
Para optimizar la aplicación, HDFS se diseñó de tal forma que la lógica de procesamiento se realiza cerca de los datos y no viceversa como se acostumbra hacer.
La arquitectura de HDFS consiste en clusters conectados compuestos por nodos que almacenan los archivos. Un cluster HDFS consiste en un único nodo conocido como **Name Node** que administra el namespace del sistema de archivos y el acceso de los clientes a éstos. Los **Data Nodes** almacenan los datos como bloques dentro de un mismo archivo.
Dentro del sistema, un **Name Node** administra operaciones como abrir, cerrar y renombrar archivos y directorios. Este tipo de nodo también se encarga de

mapear bloques de datos a los **Data Nodes** que manejan los request de escritura y lectura de los clientes. Los **Data Nodes** también se encargan de crear, eliminar y duplicar bloques de datos.
A continuación, se presenta un diagrama para poder visualizar la estructura de un sistema HDFS: