Pruebas

- Lunes; 29 Abril
- Lunes: 3 Junio
- Lunes: 1 Julio

¿Que es algo distribuido?

- Es un sistema en el que los componentes de hardware o software:
    - Se encuentran en computadores **unidos mediante una red**
    - Se comunican únicamente mediante **paso de mensajes**

Es algo que está en todos lados
Ej: ver la distribución de Tracert en CMD para ver por donde pasa la solicitud

Teorema CAPS

Características de un SD

- **Concurrencia**
    - Varios componentes acceden a la vez a un recurso compartido
    - Ej: En hardware: Impresoras, discord
    - Ej en Software: ficheros, BD..
- **Inexistencia de un reloj global**
    - Necesidad de temporalidad para coordinación
- **Fallos independientes**
    - Por aislamiento de red (Red)
    - Por parada de un computador (Hardware)
    - Terminación anormal de un programa (Software)

Desafíos

Heterogeneidad

Niveles de Heterogeneidad

- **Redes:** Distintos protocolos de red
- **Hardware:**distinta representación de datos
- **Sistemas Operativos:** distintas llamadas al sistema
- **Lenguajes de programación:** representación de estructuras de datos, métodos, etc
- **Implementaciones:** adaptación de estándares

Solución

- Middleware, extracto de Software que enmascara la heterogeneidad adyacente

Extensibilidad

Grado en que se pueden añadir y publicar nuevos servicios para su uso por una variedad de clientes.

- Ej: Publicación de interfaces

Ejemplos

- RFC (Request for Comments)
    - Propuestas de protocolos para internet
- WSDL (Web Service Description Language)
    - Descripción y publicación de servicios web SOAP

Seguridad

Comunicación por paso de mensajes

- Los mensajes pueden ser manipulados

Desafíos principales

- **Confidencialidad:** lectura de mensajes por terceros
- **Integridad:** Modificación de mensajes por terceros
- **Disponibilidad:** interferencias al intentar acceder a los recursos

Escalabilidad

Un sistema es escalable si conserva su efectividad ante un incremento de:

- Número de recursos
- Numero de Usuarios

Puntos Clave

- Control del coste de recursos Hardware
- Prevención de desbordamiento de recursos Software
    - Ej: paso de IPv4 a IPv6
- Evitar cuellos de botella/perdidas en estaciones
    - Algoritmos descentralizados
    - Replicación
    - Uso de caches

Tratamiento de Fallos

En sistemas distribuidos, los fallos siempre son parciales

- **Detección de fallos**
    - Checksum para fallos en la transmisión
        - Utilizado para verificar la integridad de la información mediante la detección de errores
    - Detección de caída de servidores
- **Enmascaramiento de fallos**
    - Servidores Proxy
    - Retransmisión de mensajes fallidos
- **Tolerancia a fallos -> Redundancia**
    - Rutas alternativas entre routers

- **Disponibilidad**
    - Refiere a que el sistema esté operativo/ en línea

Concurrencia

Cada objeto debe ser responsable de operar correctamente en el entorno concurrente

- Objetos deben implementarse para trabajar correctamente en entornos SD
    - Servidores multiHilo
    - Sincronización mediante semáforos

Transparencia

- El sistema se percibe como un todo, en vez de como una colección de componentes independientes.

¿Crees que se puede conseguir una transparencia perfecta? Es decir, ¿crees que puede ejecutarse un método remoto como si fuera local?

- Lograr una transparencia perfecta en la ejecución remota como si fuera local es difícil debido a la latencia y otras limitaciones de la red. Aunque tecnologías como RPC pueden ocultar la complejidad, siempre habrá diferencias debido a las limitaciones de los sistemas distribuidos.

Tipos de transparencia

- **De acceso**
    - Google File System
- **De ubicación**
    - Cloudflare, Akamai
- **De concurrencia**
    - Google Spanner o Apache Cassandra,
- **De replicación**
    - Amazon S3 o Microsoft Azure Storage
- **Frente a fallos**
    - Hadoop Distributed File System HDFS
- **Movilidad**
    - Kubernetes
- **Prestaciones**
    - Apache Spark
- **Escalado**
    - Amazon Web Service

Desafios

- Todos los desafíos son importantes
- Dependiendo del contexto tienen más relevancia que otros
    - Transacciones comerciales -> seguridad, fallos, concurrencia
    - P2P, DNS -> escalabilidad, fallos

Resumen

- En un sistema distribuido (SD) los componentes están unidos mediante una **red** y se comunican mediante **paso de mensajes**
- En un SD pueden convivir muchas plataformas, una **heterogeneidad** que trata de resolverse mediante **middleware**
- En algunos casos es importante diseñar SDs que sean **escalables**, es decir, toleren un incremento en el número de recursos o usuarios. Especialmente importante si la escala es Internet
- En un SD puede haber **fallos a muchos niveles** que en la medida de lo posible deben ser ocultados o tolerados
- En un SD varios procesos pueden acceder a los mismos recursos a la vez. Esta **concurrencia** debe ser identificada y tratada adecuadamente
- La **seguridad** y la **extensibilidad** son otros desafíos importantes, pero no nos centraremos mucho en ellos
- Un tratamiento correcto de estos desafíos lleva a un sistema **transparente** en el que el usuario final no percibe todos estos aspectos
- Es importante definir el **contexto** en el que nos encontramos para identificar los desafíos clave, o los que vamos a necesitar enfrentar en nuestro **modelo** de SD

Centralizado a Distribuido

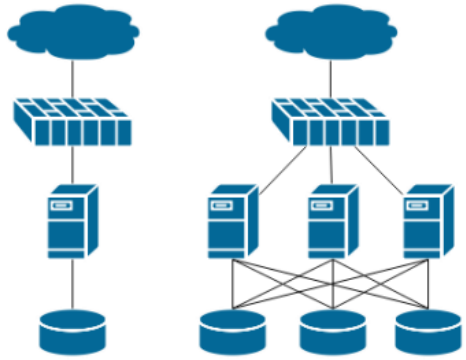
- Redes centralizada mantienen todos los datos en una unica computadora/ servidor
- Distribuida funciona como una red de datos lógica, EJ]: blockchain

Al distribuir, disponemos de dos de estos tres: Disponibilidad, consistencia y Tolerancia a particiones.

- Consistencia y Disponibilidad
- Disponibilidad y T. a particiones
- Consistencia y tolerancia a partición

No se puede tener los 3 al mismo tiempo

Centralizado VS Distribuido

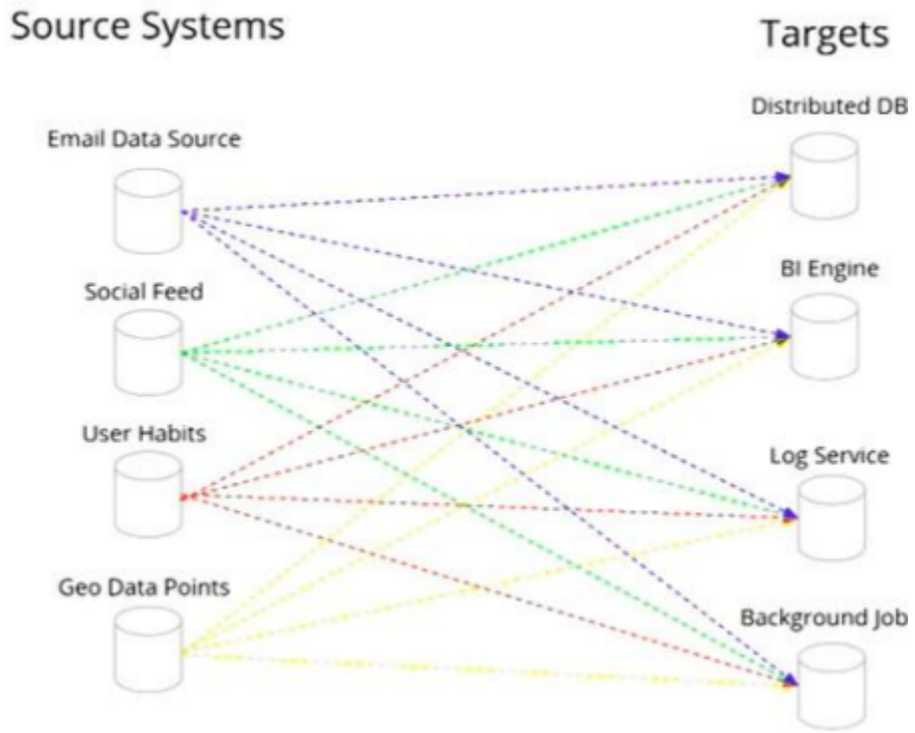


Centralizado

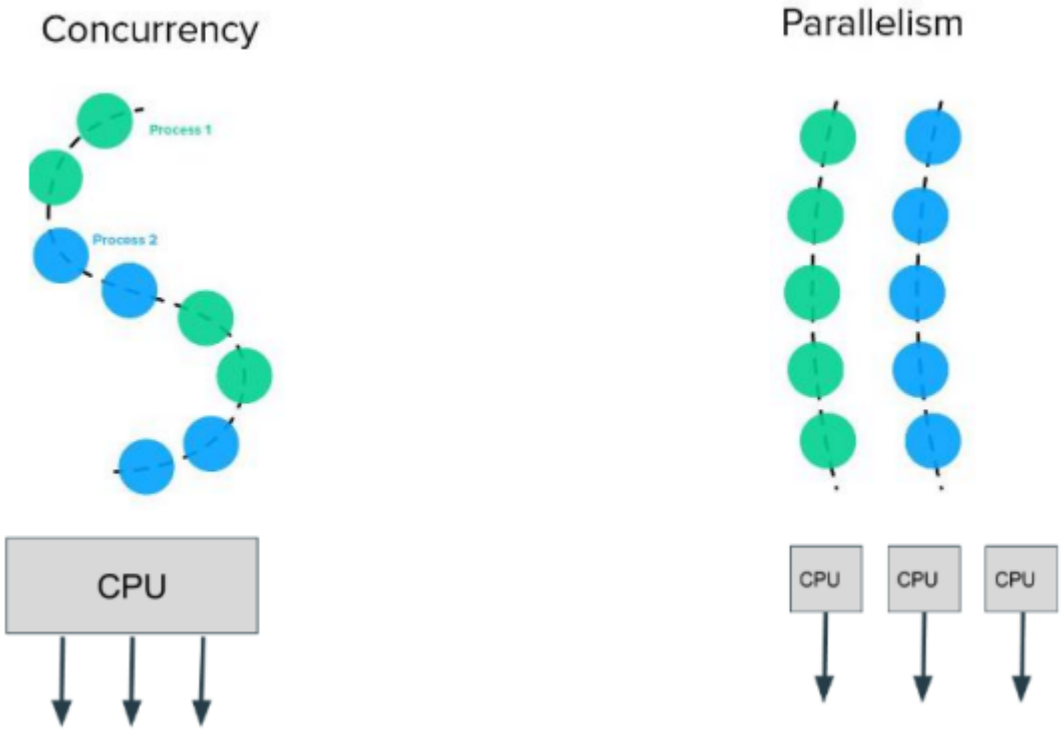
- Ventaja de gestión más sencilla, control de recursos y decisiones más eficientes.
- Mayor riesgo de fallos, falta de escalabilidad y dependencia a la entidad central

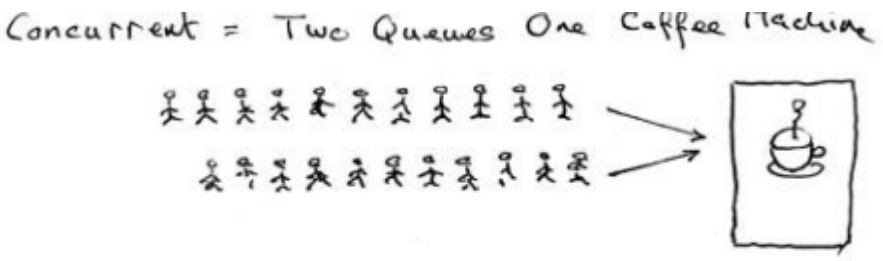
Nuevas funcionalidades de los sistemas

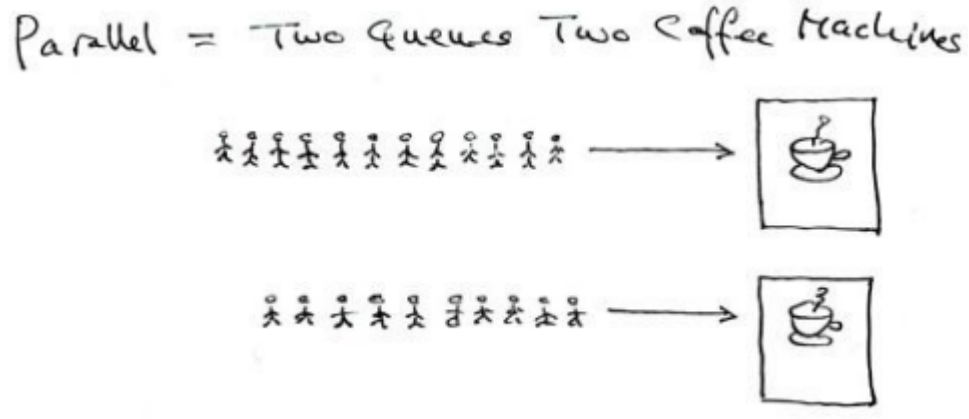
- Se requieren al menos 16 iteraciones



Concurrencia / Paralelismo







Cuando usar S. Distribuidos

- Carga masiva de archivos
- Transacciones Masivas
- Alarmas dado un evento
- Renderizar imágenes en diferentes formatos

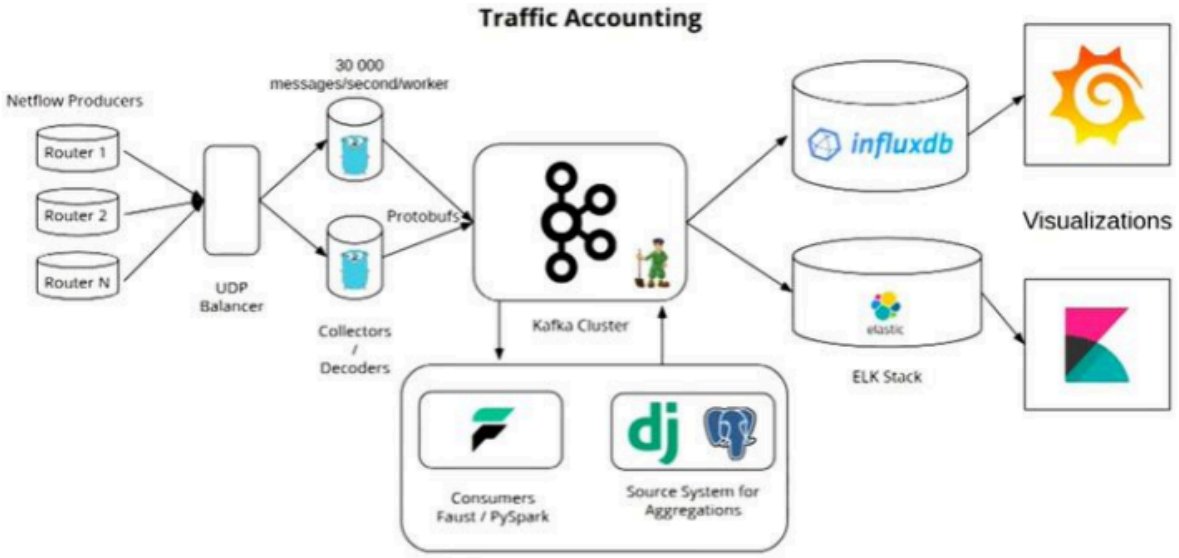
Como se construye un sistema Distribuido?

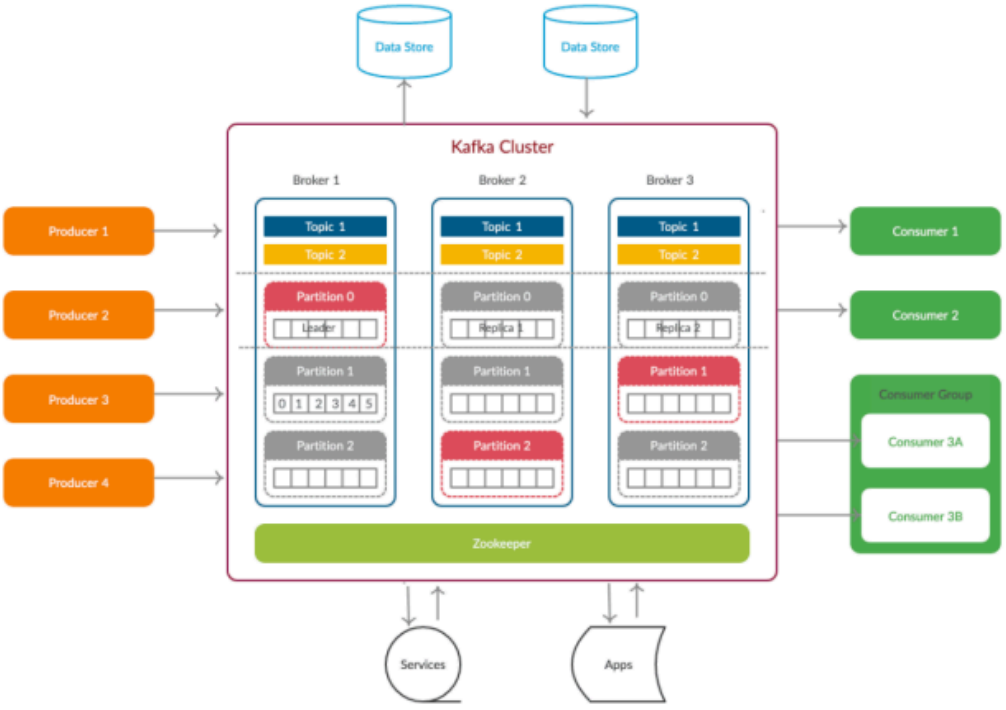
- Dividir problema en tareas
- Tratar de concatenarlas (workflow)
- Sincronizar los procesos

CELERY

- Encolador de tareas
- Codigo Asincronico
- Manejo de errores
- Distribuido
- Utilerias: Flowe, Kombu, Orquestadores
- Integracion: Django, Flask

Diseño de un SD





Zookeeper

- Es un servicio de coordinación y gestión distribuida que se utiliza ampliamente en sistemas distribuidos

Protobufs

- Es un formato de sincronización de datos desarrollado por google. Eficiente y flexible para estructurar, serializar y transmitir datos estructurados entre diferentes sistemas o aplicaciones.
    - Es una mejora a los protocolos JSON y HTTP

PySpark

- Motor de procesamiento de datos distribuidos.
- Diseñado para el procesamiento de datos a gran escala.
- Incluye procesamiento por lotes y en tiempo real mediante Spark Streaming

Ventajas de distribuir sistemas

- Economía
- Aumento de capacidad de procesamiento
- Aplicaciones inherentes distribuidas
- Capacidad de crecimiento

Desventajas

- Aumento de Complejidad
    - ¿Cómo verifico que no queden anomalías?
    - Costos de administración
- Fallas más frecuentes
    - Red, latencia, pérdida de mensajes
- Interoperabilidad
    - Falta de estándares
- Seguridad

Implicancias de LOS S.D

- Concurrencia
    - Recursos compartidos
    - Sincronización
- Sin tiempo
    - Cada PC tiene su propio reloj local
    - Coordinación
- Fallas independientes

Desafíos de los S.D

- ## Compartir recursos

Es el objetivo básico de los sistemas distribuidos

- ## Transparencia

Migración

Para nosotros un servidor debe parecer un solo lugar, no importa donde este.
Ej: dirección de la phoenix uls.
Mientras más transparencia, más se que no estoy pasándome a otro servidor

Fallas

Usuario no sabe (incluso nunca supo) que un recurso ha dejado de funcionar (o se ha recuperado)
Ej: Arreglos de discos RAID, Google File System

Persistencia

Tiene que ver con otros servidores que nos garanticen que los datos no se vayan a perder
Aplicable a distintos sistemas, no solo distribuidos

Transparencia tiene límites y grados

1. No siempre es posible Esconder todo

Ej: Desfase temporal de comunicación entre puntos muy distantes (Delay)

2. Siempre hay un trade-off entre transparencias y rendimiento

Ej: Bases de datos replicadas word-wide. Un caso peculiar son los servidores DNS que pueden tomar días en actualizarse, lo cual no se oculta al usuario

- ## Usabilidad

- ## Escalabilidad

Se puede dimensionar en 3 aspectos

Tamaño

Capacidad de agregar fácilmente más recursos y usuarios al sistema

Geografía

Capacidad de que usuarios y recursos se encuentren en distancias considerables

Administración

Capacidad de mantener una administración fácil a pesar del crecimiento

- Hardware

1. Memoria compartida - BUS
2. Memoria Independiente - BUS
3. Memoria Compartida - Switch
4. Memoria independiente - Switch

- Multiprocesadores

- Multi-computadores

- Software

Se encuentran tres variantes de plataformas de software para sistemas distribuidos

1. Sistemas operativos distribuidos
    a. Primitivas de sincronización: semáforos, etc
2. Sistemas operativos de red
3. Middle-ware

Organizaciones Típicas Cliente Servidor

- Protocolo HTTPS (familia del TCP/IP)

2 (o mas) capas

Arquitecturas Modernas

Clase 25 - 03 - 2024

La idea de distribuir, es que parezca que el servidor es uno solo.
Pasos:
Cómputo en la Nube
Contratación de Hosting

Para crear un host legal:

1. Tener un Dominio, comprado en el nic.cl
    a. Aqui se compra el nombre, y después a ese le podemos asignar la IP

Los benchmarking ya van definiendo la primera capa del modelo TCP.

- La fibra es lo que garantiza mayor ancho de banda
- Middleware permite la funcion de servidores

LAB 28-03-2024

- Sockets
- RPC
- Hilos
- RMI

04-04-2024

- RPC: manda una solicitud a otro servidor para que este responda y devuelva algún dato.
    - Lo que hace el servidor es registrar, y se genera un ciclo hasta que alguien acceda a la función

05-05-2024

- JavaRush para compilar en Java un Cliente - Servidor
- **Remote Method Invocation:**
    - middleware para que un objeto
- Servidores web:
    - Nginx
    - IS

Aplicacion RMI

- Dos fases Fundamentales
    - **Localizar objetivos remotos**
    - **Comunicarse con objetos**

RMI vs Sockets

RMI	Sockets
Invocación de objetos remotos	Invocación de métodos remotos
Sencillo	Complicado
No hay protocolo	Necesidad de un protocolo
Genera mucho tráfico	Genera poco tráfico

Implementacion RMI

Esquema

JAVA RMI

Interfaz: publicación

- Debe accesible a cliente y servidor

Middleware

Servidor P2P

- El problema de montar un servidor local P2P, es el datatime

Applet

Clusters

GRID

- Es para cálculos específicos, generalmente usados para procesos matematicos y cientificos, casi todas las maquinas deben ser idénticas (por la sincronización de relojes)

CORBA

11-04-2024 (ppt Comunicaciones)

Berkeley Sockets (JAVA)

- En los sockests, solo necesitamos desde que puerto va hasta el puerto destino

Tipos de Sockets

- Stream(SOCK_STREAM)
    - Orientado a la conexion
    - Fiable, asegura entrega del mensaje
    - No mantiene separacion entre mensajes
- Datagrama(SOCK_DGRAM)
- Raw(SOCK_RAW)

Llamadas a procedimientos remotos

Java RMI

Yo puedo acceder a una clase, también a los métodos de esta clase, y puedo enviar y recibir mensajes.

Elementos necesarios

- Codigo cliente
- Codigo servidor
- Formato de representación
- Definición de interfaz
- localización del servidor
- Semánticas de fallo

Código cliente/ Codigo servidor

Las funciones de abstracción de una llamada RPC a intercambio de mensajes se denominan resguardos (stubs).
El código cliente se refiere al programa que invoca a un procedimiento remoto, mientras que el código servidor es el programa que ofrece los procedimientos remotos para ser llamados por los clientes.
El cliente y el servidor pueden estar en sistemas diferentes y se comunican a través de la red utilizando el protocolo RPC.

Resguardos (stubs)

Son componentes de un sistema RPC que actúan como intermediarios entre el cliente y el servidor.

- En el lado del cliente, el stub proporciona una interfaz similar a la de la función local que el cliente desea invocar remotamente.
- Igual se encarga de empaquetar los parámetros de la llamada en un mensaje RPC y enviarlo al servidor a través de la re

Semántica Fallos

Se refiere a cómo se manejan los posibles errores y fallas que puedan ocurrir durante la comunicación entre el cliente y el servidor

Cliente no puede Localizar al servidor

- Ocurre cuando el cliente no puede encontrar la ubicacion del servidor debido a:
    - problemas de red
    - Configuracion incorrecta
    - Servidor offline

Pérdida de mensajes

- Puede suceder que los mensajes RPC se pierdan en la red debido a:
    - Congestión
    - Fallos en el router
    - Problemas de conectividad
- Se puede abordar con “confirmaciones de recepción”

Fallo en los servidores

- Si el servidor falla mientras se procesa una llamada RPC, el cliente puede recibir una respuesta de error

Fallos en los clientes

- Errores mientras se realiza una llamada RPC. Puede deberse a problemas locales en la máquina del cliente, como por Ejemplo: falta de recursos

Aspectos de implementación

- Protocolos RPC
    - Orientados a conexión
    - No orientados a conexión

“RPC de Sun”

Objetivo Principal

- Simplificar el desarrollo de aplicaciones distribuidas al ocultar la complejidad de la comunicación a través de la red, permitiendo que los programas se comuniquen como si estuvieran ejecutándose localmente en la misma máquina.

Clase 12-04-2024

Proxy

SIEMPRE ES NECESARIO EL PROXY
Se COMPORTA COMO INTERMEDIARIO ENTRE SERVIDOR Y CLIENTE

- Existen dos tipos:
    - Proxy Reverso (Lado del cliente)
    - Proxy Forwarding (Lado del servidor)
        - Elemento del Firewall
- Definición
    -

- Características
    - Proporciona Cache
    - Control de acceso
    - Registro de tráfico
    - Prohibir cierto tipo de tráfico
    - Mejorar el rendimiento
    - Mantener el anonimato

Forward Proxy VS Reverse Proxy

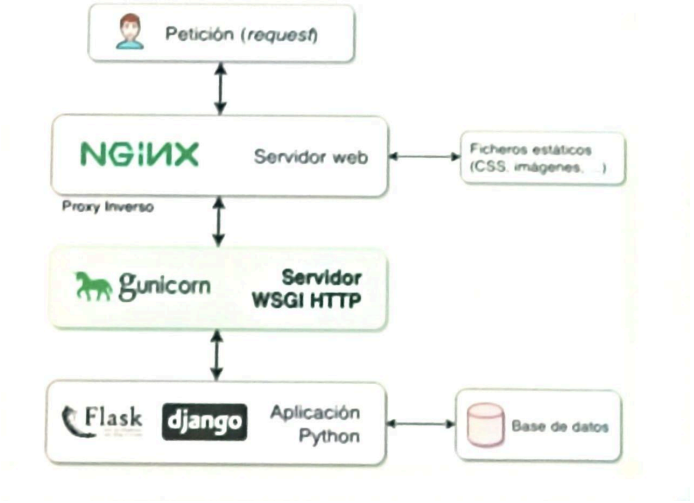
Proxy reverso de acuerdo a la IP/ruta, este redirecciona a un servidor o alguna parte del sistema

- Hoy en dia casi todo esta basado en proxy reverso
- Ej: Nginx es un Proxy Reverso

WSGI Web Server Gateway Interface

- La idea de su creación era atender a varios usuarios y adaptarse a los lenguajes populares
- Básicamente es una pasarela para pasar al servidor WEB
- Puede atender a varias peticiones al mismo tiempo
- **¿Por qué usarlo?**
    - instalamos un servidor web, del que su pega es recibir y responder.
    - Recibe la petición del cliente
    - Toma la petición y se la pasa a alguien llamado App Python
- Si recibe de multiples lugares
    - Llegan varios problemas, ya que esta app solo ha fluido correctamente en español, pero si llega en otro idioma no logra entenderlo.
    - Como pasa eso, hay que ordenar. Para esto, python creo el modo WSGI
    - **WSGI:**

- Esto mejora mucho el estándar de la aplicación



WSGI nos ayuda a

- Darnos flexibilidad, los desarrolladores pueden cambiar entre Gunicorn, uWSGI.. sin cambiar codigo,
- Escalabilidad: gracias al uso de gnuicorn podemos servir miles de peticiones resolviendo contenido

APPLET

- El Applet recuperado a distancia a través de un servidor web y se ejecuta en el lado del cliente. Debido a esto, las reglas son muy estrictas
- Ej: se tiene un servidor creado en python, la cual el cliente va a ejecutar
- ***Ninguna applet debería permitir la escritura en el disco ni la impresión***

Ejemplo

- Creamos un servidor web en flask, este servidor deberá estar escuchando peticiones del cliente… este se ejecutara en el lado del cliente para funcionar

Clase 15-04-2024

Qué es lo que hay antes de llegar a un cluster?
Que es el computo en la Nube?
Que es APPLET?
que es API?
<u>Las preguntas estan en el libro!!!!</u>

Clusters

¿Qué son?

- Es un término que se aplica a un conjunto de ordenadores unidos entre sí, normalmente por una red de alta velocidad y que se comportan como si fuesen una única.
    - Estos ordenadores trabajan de forma paralela o distribuida
- **Una máquina trabaja como Nodo-Maestro, se encarga de administrar, controlar y monitorear todas las aplicaciones y recursos del sistema**

Características

Suman recursos para obtener mejor procesamiento, con esto obtendremos

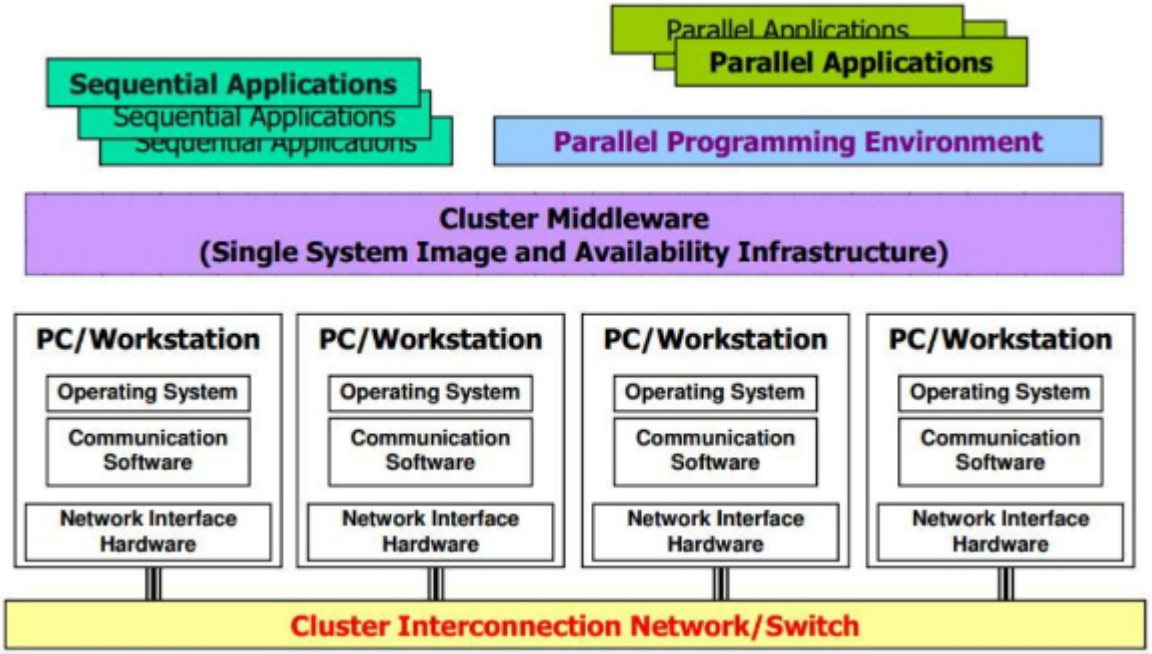
- Alto rendimiento
    - Está diseñado para dar altas prestaciones en cuanto a capacidad de cálculo
    - Los motivos para utilizar un cluster de alto rendimiento son:
        - Tamaño del problema por resolver
        - Precio de la máquina necesaria para resolverlo
    - Ej: De fontana, softland, Geovctoria
- Alta disponibilidad
    - Se caracteriza por mantener una serie de servicios compartidos y por estar constantemente monitoreado entre sí.
    - Garantiza el funcionamiento ininterrumpido de ciertas aplicaciones.l
- **Balanceo de carga**
    - Comparte el trabajo a realizar entre varios procesos, ordenadores u otros recursos
- Escalabilidad
    - Es la propiedad deseable de un sistema, una red o proceso, indica:
        - Habilidad para reaccionar y adaptarse sin perder calidad
        - Estar preparado para hacerse mas grande sin perder calidad en los servicios ofrecidos

Usos de Clusters

- **Actividades**
  - Servidores web
  - Comercio electronico
- **Cluster en Apps empresariales**
  - Flickr
  - Wikipedia
- **Clusters en aplicaciones científicas**
  - Predicción meteorología
  - Simulaciones

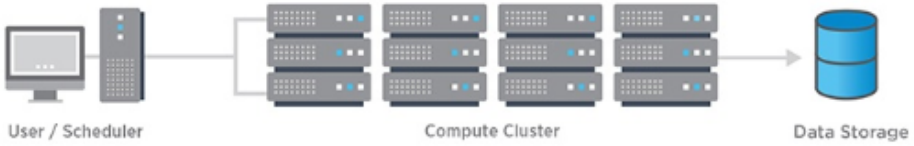
Componentes de un Cluster

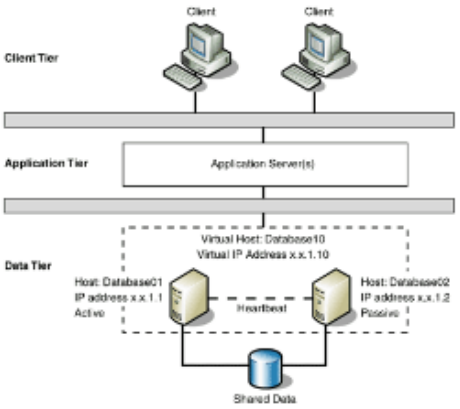
- **Nodos:**
  - Punto de intersección de varios elementos que convergen en el mismo lugar
- **Almacenamiento:**
  - NAS (A. conectado a una red), es como la nube, pero en mi casa
  - SAN (Red de área de almacenamiento)
  - Almacenamiento interno en el servidor
- **Sistema Operativo**
  - Deben ser multiproceso y multiusuario
- **Conexiones de red**
  - Los nodos de un cluster pueden conectarse mediante una simple red ethernet
- **Middleware**
  - Actúa entre el SO y las aplicaciones
  - Funciona como una capa de traducción oculta para permitir la comunicación y la administración en aplicaciones distribuidas
  - Se encarga de distribuir los datos
  - ***Es una capa de abstracción de software distribuido***
- **Ambiente de programación paralela**
  - Permite implementar algoritmos que hagan uso de recursos compartidos
    - CPU
    - Memoria
    - Datos y servicios

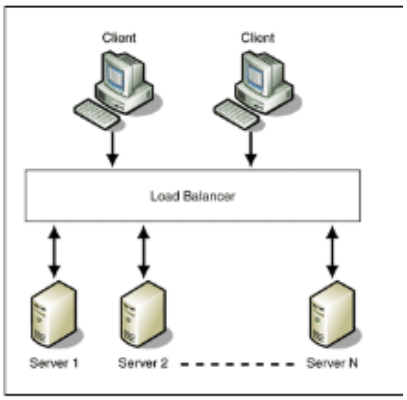


Tipos de Cluster

- **HPCC (High Performance Computing)**
  - Utilizada para cálculos complejos que requieren alta capacidad computacional y/o alta capacidad de memoria
- **HACC (High Availability)**
  - Garantiza disponibilidad continua y confiabilidad al usuario
- **HTCC (High Throughput Computing Clusters)**
  - Procesamiento rápido de grandes volúmenes de datos
    - Mayor cantidad de tareas en el menor tiempo posible
- **Load-Balancing**
  - Distribución equitativa de carga para la eficiencia







Middleware

Los servicios de middleware pueden ser representados en dos grupos

- OLE: sirve para compartir recursos

Servicios de desarrollo

Servicios de administración

ODBC

- Es uno de los middleware de windows
- Por lo general cuando uno abre un middleware de base de datos, si soy el administrador, lo agrego al sistema.
    - Si lo leo de usuario, solo el podra tener acceso
    -

Para armar un cluster

Centralizado vs Distribuido

Proxy Reverse

Proxy Forwarding

¿Cuándo usar Sistemas Distribuidos?

- Renderizar imágenes en diferentes formatos
- Para generar muchos reportes “Complejos”
- Carga masiva de archivos

Como se construye?

1. Dividir el problema en tareas pequeñas
2. Tratar de concatenar estas tareas (**Workflow**)
    a. **WORKFLOW:** se usan en los sistemas modernos,
3. Sincronizar los procesos

CELERY

- Encolador de tareas
- Manejo de errores
- Código asíncrono
- Distribuido
- Utilerias: Flower, Kombu

¿Cómo se diseña?

1. Cluster de Kafka

Protobuf

Es un formato de serialización de datos desarrollado por google. Forma eficiente y flexible de estructurar, serializar y transmitir datos estructurados entre diferentes sistemas o aplicaciones .

- Es una mejora a los protocolos JSON o HTTP

Sincronización

- Lo único que no puede compartir un sistema distribuido es el clock y la memoria
- Los procesos intercambian mensajes sobre un canal de comunicación donde el retardo es impredecible.

Consecuencias

- Las nociones tradicionales de tiempo y estado no funcionan
- Ausencia de un reloj

Desafíos

- Sincronización de relojes
- Evitar DeadLock
- Si un nodo falla: Algoritmos de elección

Sincronización de relojes

Soluciones

- Algoritmo de Christian

Solucion Christian

- Algoritmo de sincronización externa basado en el RTT
- Para mejorar la precisión se puede repetir el algoritmo varias veces y descartar las que tengan un RTT superior al límite
- Redes más rápidas -> menor RTT -> Mejor medición
- Una estimación de tiempo de propagación sería

Algoritmo de Berkeley

- Se realiza periódicamente
- S

Protocolo NTP

- Chris y Berkeley son adecuados para redes rápidas (Principalmente usadas en intranet)
- La sincronizacion entre cada par de elementos de la jerarquia
    - Modo multicast
    -
    -
- Mills en 1995, definió arquitectura y prototipo
- Entonces bastaría conectarse con NTP y ya todos estaríamos coordinados

Solucion Lamport

Relojes Lógicos (Lamport)

- Según Lamport, la sincronización de relojes no debe ser absoluta, debido a que si dos procesos no interactuan entre si, no requiem que sus relojes están sincronizados
    - Un ejemplo de 3

Vectores de Relojes Lógicos (Mattern y Fidge)

Uso de Relojes Lógicos

- La utilización de relojes lógicos se aplica a:
    - Mensajes periódicos de sincronización
    - Campo adicional en los mensajes intercambiados
- Por medio de relojes lógicos se puede resolver:

Exclusión mutua Distribuida

Mecanismo de coordinación entre varios procesos concurrentes a la hora de acceder a recursos/secciones compartidas

- La solución definidas para estos problemas:
    - Algoritmos Centralizados
    - “” Distribuidos

Exclusión Mutua

Exclusión Mutua “Centralizado”

Rendimiento

- Ancho de banda
    - Si hay poca banda de ancha, no se podra atender a todos.
    - Acceso al recurso implica dos mensajes
- Retardo del Cliente
    - **enter()**
    - **exit()**
- Throughput (Rendimiento)
    - La finalización de un acceso a la región crítica implica un mensaje de salida y un **OK** al siguiente proceso en espera
- Tolerancia a fallos
    - La caída del elemento de control es crítica
    - La caída de los clientes o la pérdida de mensajes se puede solucionar por medio de temporizadores

Generales Bizantinos

- **Error Bizantino**: Un proceso genera valores de forma arbitraria

Transacciones Concurrentes

Conceptos

- Phoenix funciona como vista-controlador, pero para el usuario parece cliente-servidor
- Middleware: es una capa sobre el sistema operativo
- OMA: arquitectura que administra los objetos
- URP: arquitectura donde se almacena objetos
- API REST
    - GET
    - PUT
    -
- Gunicorn
    - Es el servidor HTTP WSGI más usado en python
- **ERP:**
    - Ejemplo:
        - GeoVictoria (Control de asistencia)
            - Lleva un control de donde esta el personal de la empresa contratante, osea muestra donde se encuentra el personal.
        - Softland Cloud
- Mejor base de datos: oracle, le sigue: My SQL

#No hacer ERP de RECURSOS HUMANOS

CORBA

- Paradigma orientado a objetos
- Sistemas distribuidos orientados a objetos
- Nucleo de **ORB**

Red de Superposición

- UTORRENT: instala una capa superior que permite detectar a quienes estaban en esta red superpuesta, aqui se podían buscar recursos que uno necesitara

APPLET

- Flash: Se ejecutan en otro contexto, pero se les podia enviar parametros para interactuar.
- Se descarga el applet y este corría en el servidor
- Se usaron como encuesta

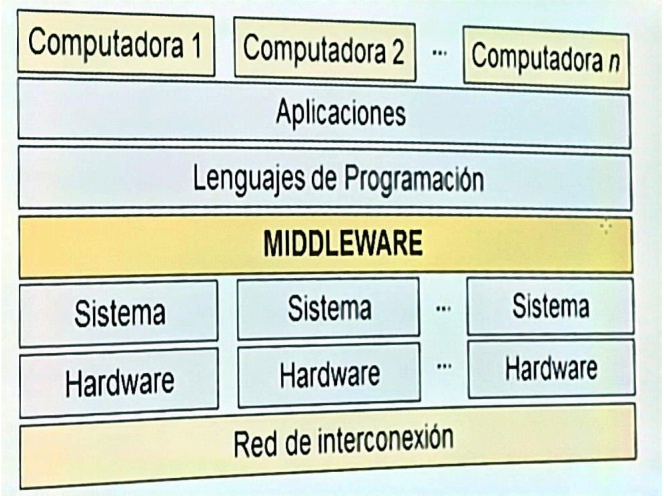
Arquitectura de Capas

Las capas que uno quiera construir, depende de cada uno, de las mas populares construidas se llama **Modelo Vista-Controlador**

Middleware

- Es un programa que está sobre el sistema operativo

Hace parecer que estamos todos juntos.
Ej: Todos veían la pantalla principal/admin



Como podriamos saber si es un cluster?
Que pasa cuando se le hace Ping a google? es señal de cluster o no?

Referido a Amazon

Problema: lo gratuito esta limitado a espacios muy pequeños

- Los cluster hacen escalar mas facil la maquina, pero eleva su costo