Escuela de Ciencias de la Computación
Final 2025-I
CC4P1 Programación Concurrente y Distribuida

Implementación de un Sistema Distribuido para el entrenamiento de IAs que requiera entrenamientos paralelos, concurrente y distribuido de redes neuronales para entrenamiento, y Consumo de Modelos entrenado de las redes neuronales de IA mediante el algoritmo de consenso RAFT.

El objetivo de esta evaluación es diseñar e implementar un sistema distribuido que permita el entrenamiento y consumo de modelos de inteligencia artificial (IA) de manera paralela, distribuida y concurrente. Se debe garantizar la gestión adecuada de los modelos entrenados y su posterior consumo utilizando el algoritmo de consenso Raft.

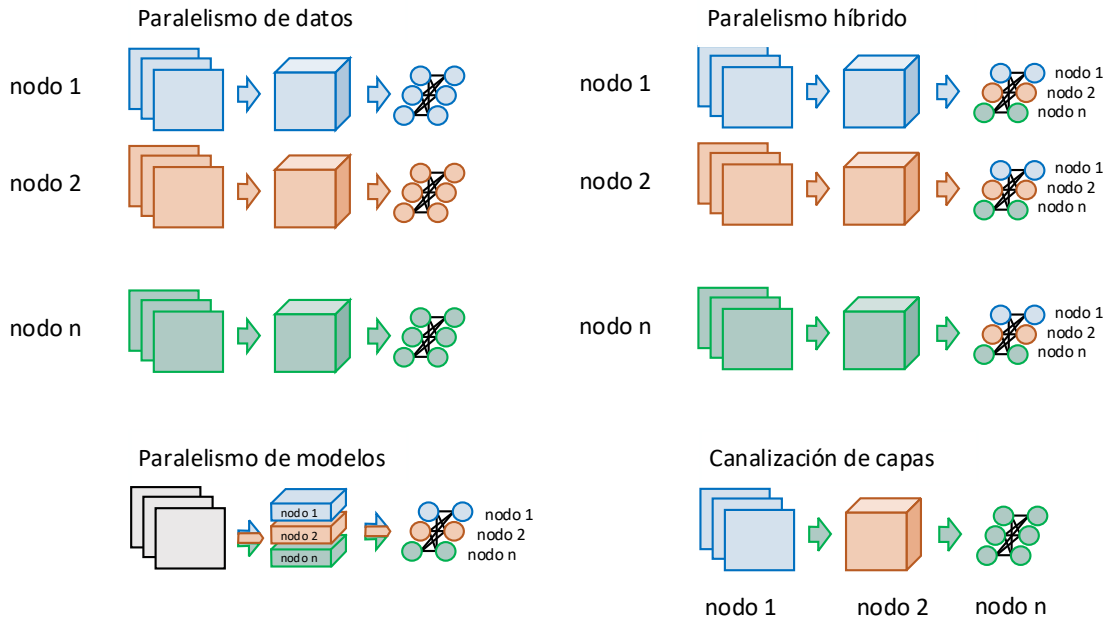
Fase 1 (Entrenamiento): Entrenamiento Distribuido de Modelos de IA

Se debe desarrollar un sistema distribuido en el cual un cliente envíe datos de entrada (inputs) y salida (outputs) a un conjunto de servidores (nodos). El modelo puede usar imágenes o con texto. Estos servidores trabajarán en paralelo, concurrente y distribuido para entrenar modelos de IA, gestionando los recursos de manera distribuida y concurrente.

Requisitos:

- Cada vez que el cliente envíe datos de entrenamiento, el sistema debe asignar un identificador único al modelo.
- La carga de trabajo debe distribuirse entre los nodos del sistema para optimizar el proceso de entrenamiento.
- Se debe garantizar la persistencia y accesibilidad de los modelos entrenados para su posterior consumo.

El grupo puede usar cualquier método de paralelismo, concurrencia para su modelo de IA con redes neuronales, se le plantea opcionalmente las técnicas de paralelismo de datos, paralelismo híbrido, paralelismo de modelos, canalización de capas u otro que el grupo crea conveniente.



El testeo puede ser secuencial o en un solo nodo como base, pero todo dependerá de programación y desempeño de su código.

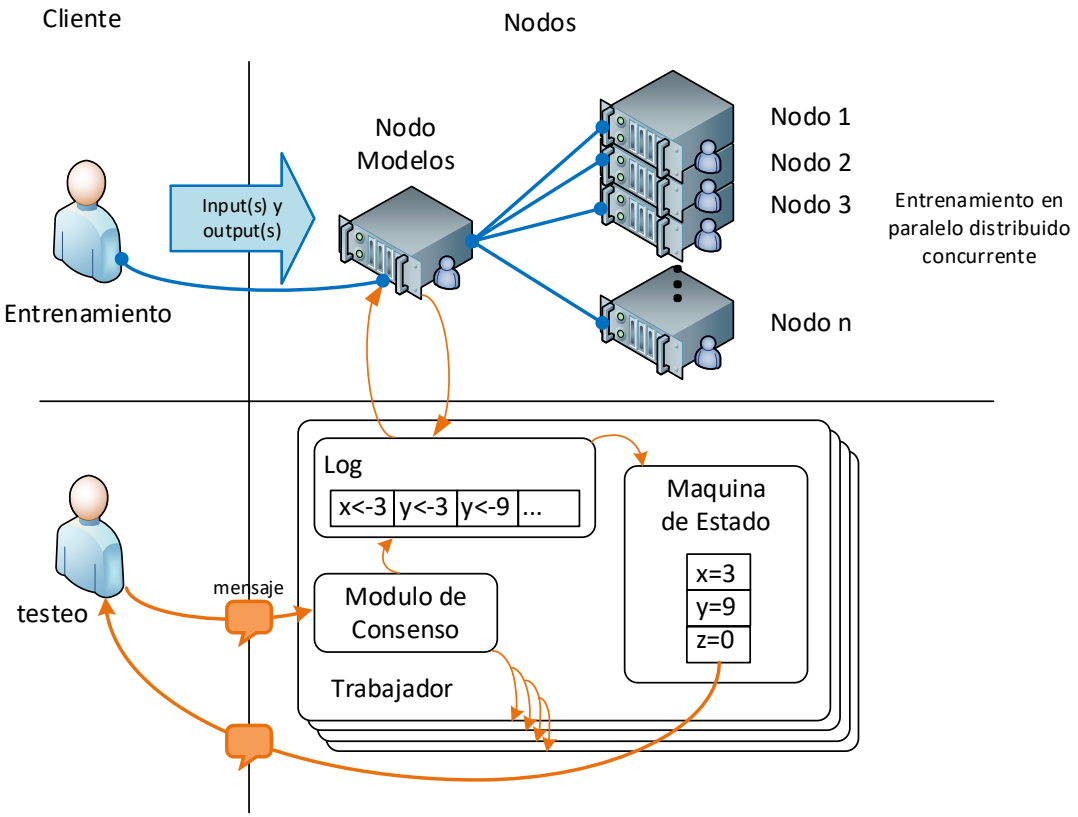
Para el caso de un nodo, este nodo n debe tener varios núcleos, todos los núcleos de un nodo se deben de usar para el entrenamiento.

Fase 2 (Testeo): Consumo de Modelos de IA con Algoritmo de Consenso Raft

En esta fase, el sistema debe permitir que un cliente seleccione un modelo de IA previamente entrenado y envíe datos de entrada para obtener respuestas del modelo. Para asegurar la coherencia y disponibilidad del servicio, se implementará el algoritmo de consenso Raft.

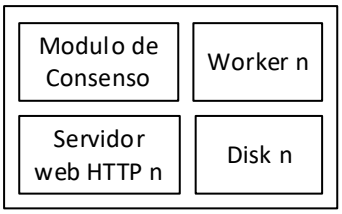
Requisitos:

- El cliente debe poder seleccionar un modelo de IA por su identificador.
- Los nodos del sistema deben coordinarse mediante Raft para garantizar la consistencia en la ejecución de consultas.
- El modelo debe procesar la entrada y devolver la salida esperada de manera eficiente.

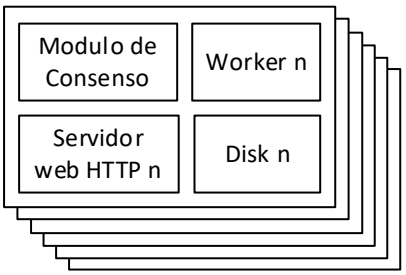


Workers (Trabajadores)

Los workers n tienen embebido un servidor web para poder administrarlo y el módulo de consenso. El módulo de consenso escoge y tiene el líder de los workers que tiene el algoritmo de consenso. El Servidor web tiene por objetivo mostrar el monitor del respectico worker n.

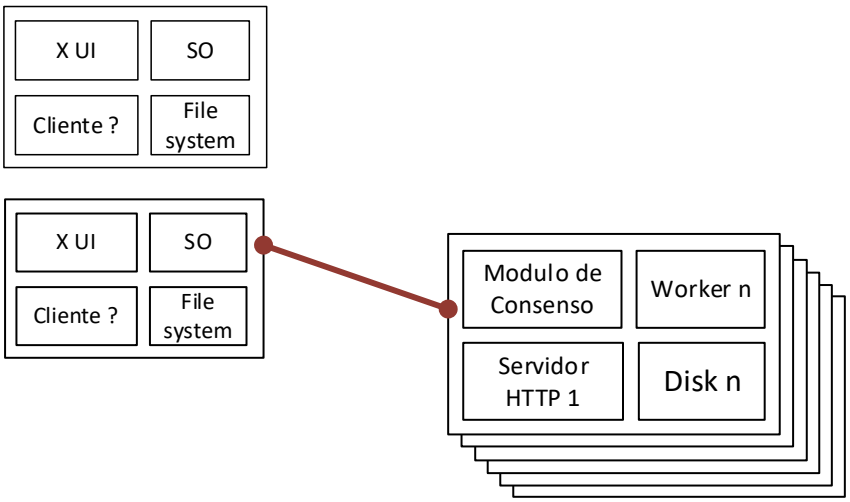


Se harán replicas de los archivos compartidos con el algoritmo de consenso.



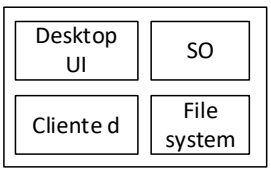
Cliente

Los clientes son desktop se conectará con el worker líder escogido mediante el algoritmo de consenso. Las funciones del "Cliente ?" es poder enviar inputs y recibir outputs de la transacción.

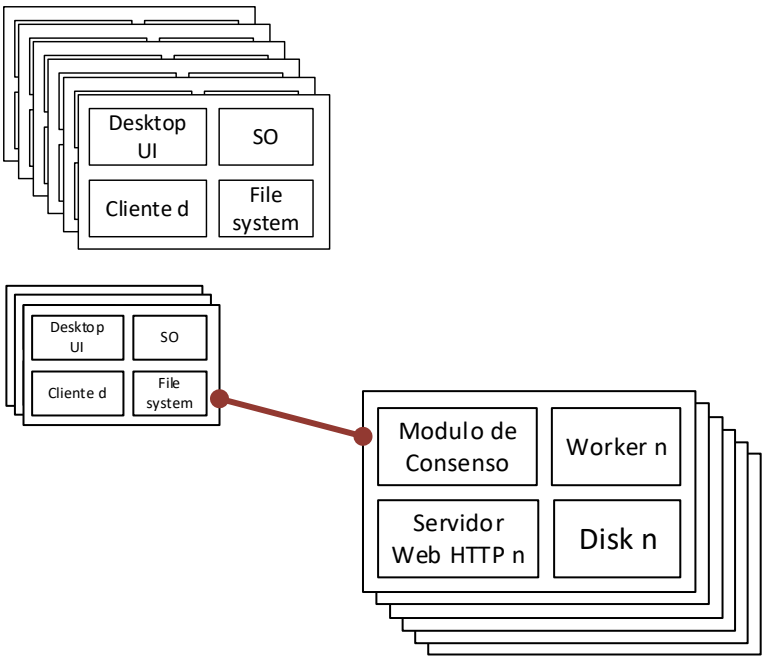


Los clientes son desktop.

Cliente "Desktop d" se puede ejecutar



Pueden tener varios clientes desktop que suben, consultan y descargan archivos.



- Se solicita como mínimo un lenguaje de programación, y luego se puede adicionar más lenguajes de programación en = {LP1, LP2}, exponer y redactar un informe, por cada alumno adicional se agregará uno más {LP3, LP4, ...}, los lenguajes de programación adicionales se tomarán encuentra para la evaluación.
- Se iniciará los nodos “workers n” luego se inician los clientes y recibirá las peticiones de Clientes.
- Sólo el módulo del entrenamiento de IA como mínimo puede estar solo en Java versión mínima 8, los lenguajes de programación adicionales se tomarán encuentra para la evaluación.
- Tomar como base las explicaciones y el código de clase.
- Usar como base las librerías de desarrollo base que viene con el lenguaje de programación base.
- Exponer y Ejecutar en cluster, con = {LP1, LP2, …}, para poder verificar el consenso.
- Para validar la consistencia de los archivos, se podrán visualizar en los monitores de servidores web de los “workers n”.
- Sistema Operativos, SO1 y SO2, donde SO1 <> SO2 y SO1 = SO2 = {LP1, LP2, …}.
- Graficar la arquitectura diseñada.
- Graficar el diagrama de protocolo.
- Usar solo Sockets de cada Lenguaje de programación.
- No usar websocket, socketio, frameworks, RabbitMQ, MQ, Librerías de Comunicación, etc.
- Explicar el desarrollo del programa puntualmente.
- Desplegar el programa en redes LAN y WIFI.
- Evaluar el desempeño haciendo un script donde el cliente ingrese aleatoriamente 1000 nuevos archivos o más, para evaluar desempeño del algoritmo de consenso.
- Poder ingresar cada cliente, los registros uno por uno, y visualizarlos en los monitores de los “workers n”.

- Como base en los “workers n” realizar el código en terminal y sus monites puede ser visualizados de otra PC con SO que tenga interfaz gráfica. Los clientes pueden tener interfaz gráfica para poder subir, listar y descargar archivos.
- Usar hilos para mejorar el desempeño y evitar corrupción de registros.

Subir en Univirtual.

- Comprimido consta
    - Códigos fuente de extensión en el LP1, LP2, LP3, ... (subir solo códigos fuente o explicativos como base, no subir otros archivos como historial, motor del lenguaje de programación u otros no necesarios)
    - PDF Informe.
    - PDF Presentación.
    - Los grupos mayores a 2 alumnos desarrollaran un lenguaje de programación adicional LP3, … para otro nodo.