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

Desarrollar un proyecto Final ORIGINAL. Debe formar un grupo alumnos para colaborar en el proyecto. Al final del semestre, entregará su código y una breve descripción del diseño y la implementación de su proyecto, y hará una breve presentación en clases sobre su trabajo.

El documento debe tener una presentación y un informe como mínimo unas 4 páginas de texto que nos ayuden a comprender qué problema resolvió y qué hace su código.

Construcción de una librería distribuida, concurrente y tolerante a fallos para estructuras de datos distribuidos y concurrentes como array, entero y double sin uso de frameworks externos

Objetivo

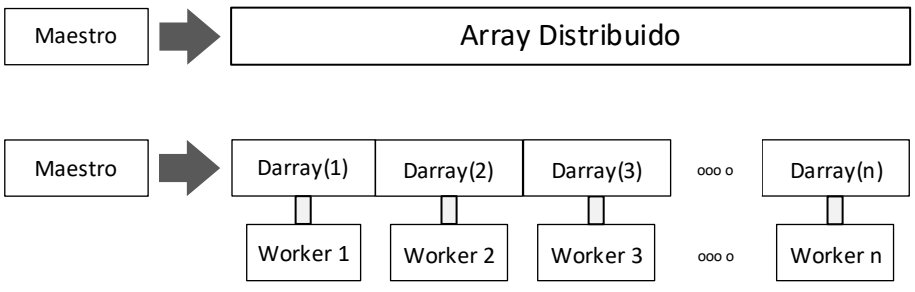
Elaborar una librería propia (en uno o varios lenguajes de programación como Java, Python o Node.js) que permita el procesamiento concurrente, paralelo y distribuido de arrays de enteros (int) y decimales (double), con mecanismos propios de resiliencia (replicación, recuperación, supervisión de nodos), y con uso exclusivo de sockets, hilos nativos y estructuras básicas del lenguaje.

La librería debe incluir:

Tipos: DArrayInt y DArrayDouble, que soporten:

- Segmentación automática y manual de arreglos entre varios nodos.
- Comunicación entre nodos usando sockets TCP.
- Procesamiento paralelo por núcleo utilizando hilos (threading).
- Tolerancia a fallos con:
    - Replicación activa de fragmentos.
    - Detección de nodo caído (heartbeat o watchdog).
    - Recuperación automática mediante réplicas.

Ejemplo 1: Procesamiento Matemático Paralelo.



Descripción:

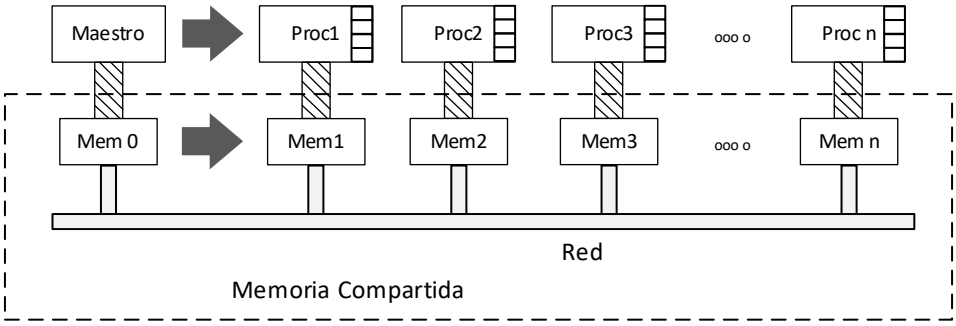
Aplicar una operación matemática compleja con un For, a cada elemento de un DArrayDouble de tamaño ≥10,000 distribuido en 3 o más nodos, ejem:

resultado = ((sin(x) + cos(x)) ^ 2) / (sqrt(abs(x)) + 1)

Condiciones:

- Usar bucles tradicionales (for) con hilos en cada nodo.
- Mostrar evidencia de ejecución concurrente.
- Consolidar resultados en el nodo maestro.

Ejemplo 2: Evaluación Condicional Distribuida y Resiliente.



Descripción:

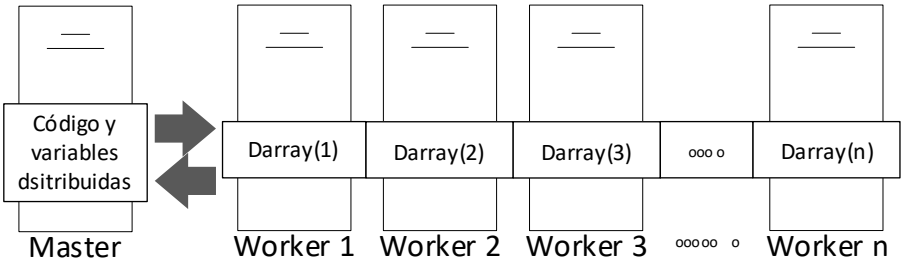
Sobre un DArrayInt distribuido, aplicar una o más condicionales en un For, ejem:

Si x es múltiplo de 3 o está entre 500 y 1000 → aplicar: (x * log(x)) % 7

Condiciones:

- Ejecutar la transformación distribuida y concurrente.
- El sistema debe continuar funcionando si uno o varios hilos internos fallan (ej. lanzar excepción), mostrando resiliencia local.
- Consolidar resultados en el nodo maestro.

Ejemplo 3: Simulación de Fallo y Recuperación Distribuida



Escenario:

Durante una operación de for distribuido en DArrayDouble, uno de los nodos se desconecta.

Requerimientos:

- Detectar la caída del nodo (por heartbeat o timeout o falta de respuesta).

- Activar un nodo con réplica de respaldo para recuperar los datos y continuar la operación.
- El resultado final debe ser correcto como si el nodo no hubiera fallado.

Entregables Esperados:

- Logs de actividad y errores.
- Evidencia de recuperación de datos desde la réplica.
- Resultados integrados y verificados.
- Consolidar resultados en el nodo maestro.

Se le pide lo siguiente:

- Diseño modular y extensible de la librería.
- Uso correcto de sockets e hilos nativos.
- Gestión de concurrencia local y sincronización, usando todos lo núcleos de nodo o worker.
- Procesamiento matemático paralelo y distribuido.
- Evaluación condicional con recuperación local.
- Tolerancia a fallos y recuperación completa.
- Se pide escribir un código, exponer y redactar un informe de mínimo 4 hojas.
- Exponer y Ejecutar en clúster para poder realizar la comparación (n nodos o tiempo) y resultados.

Presentación:

- Se pide escribir un código en = {LP1, LP2}, exponer y redactar un informe, por cada alumno adicional se agregará uno más {LP3, LP4, ...}.
- Se iniciará los nodos “workers n” luego se inician los clientes y recibirá las peticiones de Clientes.
- Ejecutar todo el programa en todos lo nodos con mínimo en java 8, luego por cada integrante agregar un lenguaje de programación adicional.
- Tomar como base las explicaciones y el código de clase.
- Exponer y Ejecutar en cluster, con = {LP1, LP2, …}
- Sistema Operativos, SO1 y SO2, donde SO1 <> SO2 y SO1 = SO2 = {LP1, LP2, …}. Según se tenga pcs con diferentes SO.
- Graficar la arquitectura diseñada.
- Graficar el diagrama de protocolo.
- Usar solo sockets e hilos 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 “array” distribuidos gigantes que puedan estar en todos los nodos del cluster y hacer pruebas, para evaluar desempeño de tiempo de ejecución de sus ejemplos.

- Poder ingresar cada cliente al ingresar el programa con tipos de datos distribuidos elaborados, y visualizarlos en los monitores de los “workers n” para observa el uso de la memoria.
- Como base en los “workers n” se ejecutar su código y en el maestro se ejecuta el programa principal.
- 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, ...
- PDF Informe.
- PDF Presentación.
- Los grupos mayores a 2 alumnos desarrollaran un lenguaje de programación adicional LP3, … para otro nodo.