Introduccion al HPC

Examen 5: Pipeline

*Merino Ortega Angel Nahum

***Abstract*— In this paper we will discuss the Pipeline design pattern, also known as the pipeline pattern, is a software design pattern that organizes and structures the execution of a set of processes or steps in a sequential flow. This pattern facilitates modularity and code reuse by breaking down a complex task into a series of smaller, more manageable steps.**

Palabras Clave- problemas, patrón, diseño, tubería, tareas, procesadores.

I. INTRODUCCION

En este documento analizaremos el patrón de diseño Pipeline, también conocido como patrón de tubería, es un patrón de diseño de software que organiza y estructura la ejecución de un conjunto de procesos o pasos en un flujo secuencial. Este patrón facilita la modularidad y la reutilización del código al dividir una tarea compleja en una serie de pasos más pequeños y manejables.

II. PLANTEAMIENTO DE PIPELINE

Se plantea lo siguiente en el patrón de diseño pipeline

Las tareas se resuelven una tras de otra.

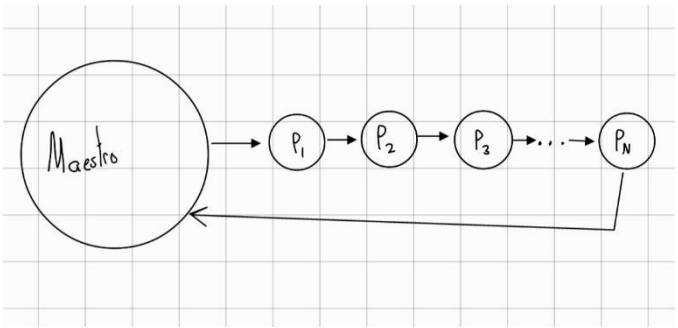
Usa como base la programación secuencial pero cada tarea es realizada por un nodo diferente.

Cada procesador ejecuta una fracción del algoritmo diferente usando la descomposición funcional.

Para llegar a la máxima eficiencia se debe llenar la tubería y en cada ciclo de reloj obtener un resultado.

Requiere que lo datos o procesamiento de la información tenga el mismo tiempo de ejecución.

Para este examen nos basaremos en la siguiente versión del pipeline:



III. CÓDIGOS GENERADOS

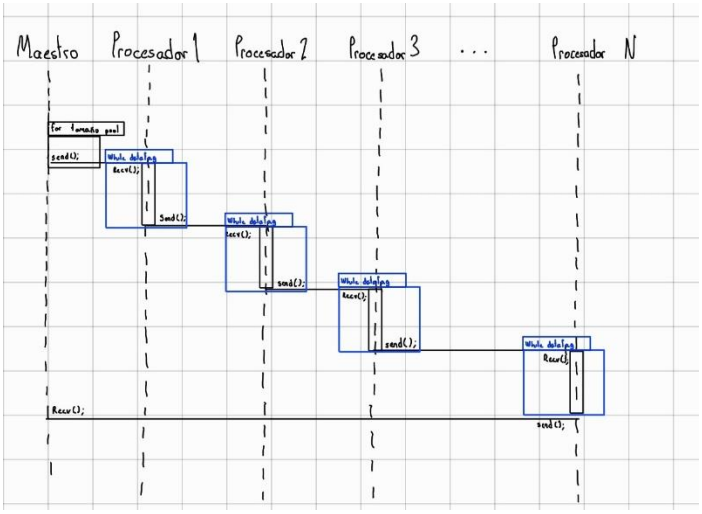
Para la mejor visualización de las diferentes acciones realizadas se generaron 2 seudocódigos en clase qué nos ayudarán a guiarnos a través de la realización de el diagrama UML.

```
Master.c
Int pool_
Int pool_datos;
For (i=0;i<tamañopool;i++){
        Send(pool_datos,p[1],”data tag”);
}
Send(NULL,P[1],”terminartor tag”)
Recv(Pool resultado,Pn,tag)
If (tag==”resultado final”{
        Usar_resultados();
}

Esclavo.c
Int dato;
If(id==pn){
        Int pool_resultados;
}
Recv(dato, procesador_anterior,source_tag);
While(source_tag==data tag){
        Resultados = procesar(dato);
        If(id==pn){
                Pool_resultados.add(resultado);
        }
        Else{
                Send(resultado,procesador_siguiente,”data
        tag”);
        }
        Recv(dato,procesador anterior,source_tag);
}
If (id==pn){
        Send(Pool_resultados, procesador, “Resultado final”);
}
Else{
        Send(null,psig,terminator_tag);
}
Dead();
```

IV. DIAGRAMA UML

Para la solución de este problema se propone lo siguiente:.



CONCLUSION

En este examen vimos el patrón de diseño Pipeline y basado en lo visto en clase y apuntes creamos el diagrama UML del patrón de diseño Pipeline o tubería. El cual nos puede ayudar a gestionar diferentes tareas que requieran varias acciones secuenciales o esperando que se termine una tras de otra para avanzar.

REFERENCIAS

Iluwatar. (n.d.). Pipeline. Java Design Patterns. https://java-design-patterns.com/patterns/pipeline/

Salesforce. (2022). Pipeline de ventas: ¿qué es y cómo gestionarlo? https://www.salesforce.com/mx/resources/articles/sales-pipeline/

Santos, H. H. (2021). Patrón de Diseño Pipeline en Laravel. DESARROLLO DE SOFTWARE EN LARAVEL. https://herminioheredia.com.mx/2021/06/08/patron-de-diseno-pipeline-en-laravel/

Santos, H. H. (2021). Aplica el Patrón Pipeline en Laravel. DEV Community. https://dev.to/herminioheredia/aplica-el-patron-pipeline-en-laravel-45ef