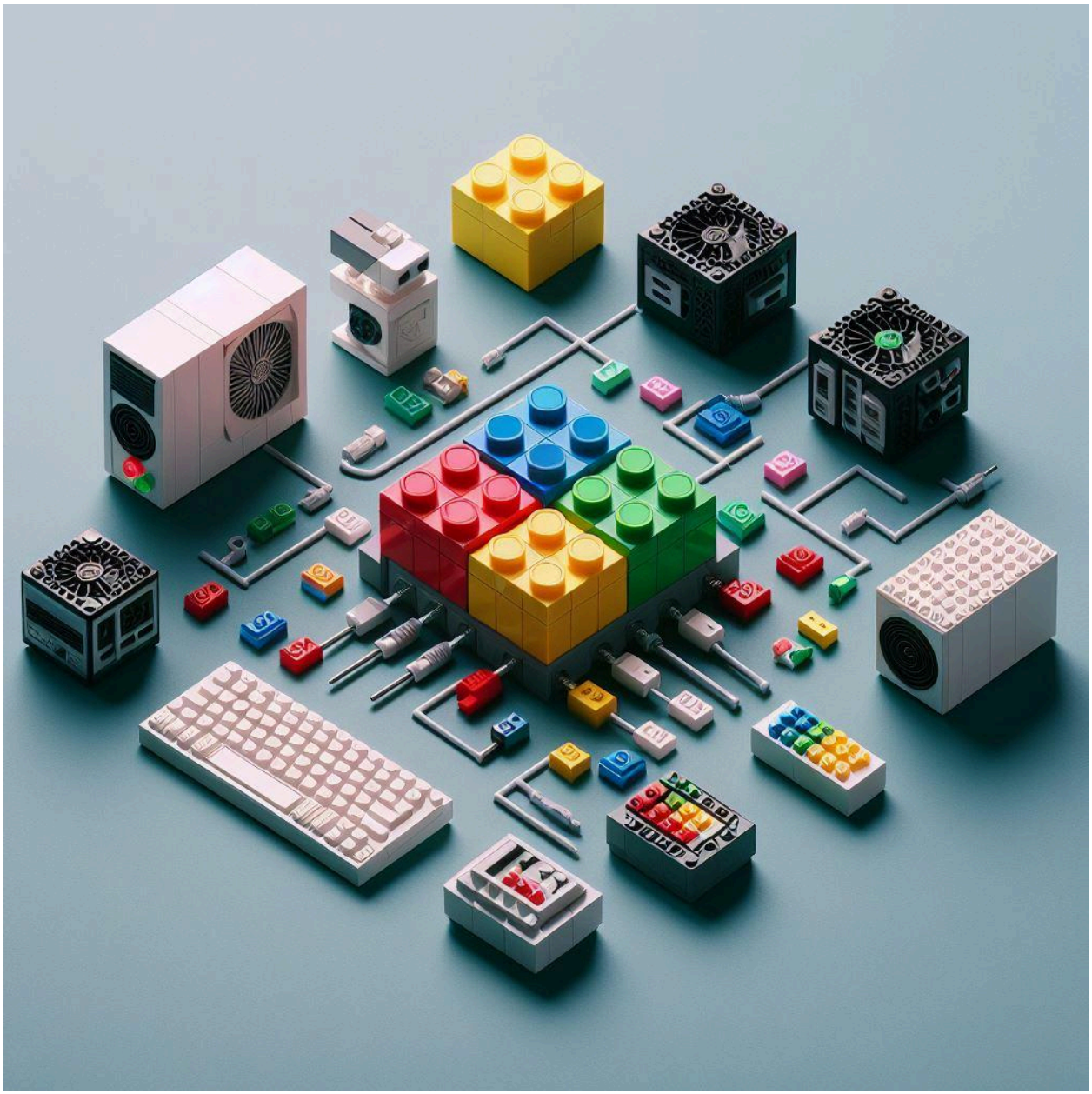
d8parti: IBM Mainframe Batch

Cómo ejecutar procesos Batch en una arquitectura open



Introducción

La arquitectura IBM mainframe tiene sus orígenes en una época, en la que el principal tipo de procesamiento era mediante la ejecución de cadenas Batch.

Es por tanto, un componente fundamental en este tipo de arquitecturas. Aunque en los últimos años/décadas el foco del desarrollo, en la mayoría de las instituciones financieras, ha estado centrado en implementar procesos Online (sistemas de medios de pago, internet banking, mobile banking, etc.), sigue existiendo un elevado número de procesos Batch de importancia crítica para el negocio.

Actualmente son muchas las entidades que intentan adoptar nuevos modelos de procesamiento, por ejemplo arquitecturas asíncronas basadas en eventos, sin embargo el procesamiento Batch o por lotes sigue siendo un modelo extremadamente eficiente que no debe descartarse y que forma el núcleo de los procesos contables en la mayoría de las instituciones financieras de mediano/gran tamaño.

¿Procesan realmente las entidades financieras la información en tiempo real?

Para entender la relevancia del Batch, hay que dar un salto en el tiempo, a los orígenes de la adopción de la tecnología por parte de las instituciones financieras. Hasta finales del siglo XX las cargas de trabajo en un entorno Mainframe se podían dividir en dos tipos, con una ventana de ejecución independiente:

- Ventana Online (horario de apertura de las oficinas). En la que los recursos de la máquina se asignan prioritariamente a los monitores transaccionales (CICS/IMS) y se restringe la ejecución de procesos Batch
- Ventana Batch. Cierre de oficinas, la capacidad se asigna a los procesos Batch diarios

El cierre de las oficinas suele estar asociado a un cambio de sesión contable y al lanzamiento de procesos Batch con acceso masivo a los datos (lectura/actualización). Este modelo permite compartir sobre la misma infraestructura Hardware ambos tipos de proceso, evitando problemas de concurrencia Online Batch (excesivo consumo de recursos hardware, bloqueos en los datos por actualizaciones masivas, inconsistencias en el acceso a datos, etc.)

Desde las oficinas se realizan las operaciones básicas de servicing de los productos y está información se consolida posteriormente mediante procesos Batch (riesgos, contabilidad

financiera/analítica, reporting, etc.).

La ejecución exitosa de dichos procesos durante la ventana Batch es crítica para la apertura de las oficinas al día siguiente.

¿Qué problemas presenta este modelo?

La tecnología Mainframe ha sido clave en el éxito de la mayoría de las entidades del sector financiero, sin embargo con el inicio del siglo y la generalización en el uso de internet, el anterior modelo de procesamiento (dos ventanas independientes Online y Batch) comienza a diluirse.

Las constantes iniciativas para la digitalización del sector (banca móvil, apificación procesos, gestión documental, analítica datos, etc.) aceleran exponencialmente la desaparición de la ventana Batch;

- La ventana Online ya no está limitada al horario de apertura de oficinas, el cliente exige un horario de atención 24x7
- Los procesos Batch deben revisarse para evitar contenciones que impidan el correcto funcionamiento de los procesos Online (descarga y proceso desde ficheros secuenciales, frecuencia de commit a la Base de Datos, etc.)
- Los datos se descargan o replican a otras plataformas analíticas, sin embargo el grueso de los procesos contables siguen residiendo en el servidor Mainframe
- Los recursos del sistema (CPU, Memoria, etc.) deben compartirse al ejecutarse todos los procesos Batch y Online sobre la misma plataforma Hardware

Como consecuencia, la complejidad y duración de los procesos Batch (Elapsed Time) tiende a aumentar poniendo en riesgo la operativa diaria de las entidades.

Las entidades se enfrentan a un círculo vicioso de crecimiento de la infraestructura Mainframe al no poder separar/aislar cargas de trabajo en servidores dedicados.

¿Cómo poder mejorar el rendimiento de los procesos Batch en una plataforma altamente distribuida basada en Kubernetes?

Como cualquier otro proceso, el rendimiento de un proceso Batch está determinado por el uso de los recursos del sistema, estos son principalmente;

- Ciclos de CPU

- Memoria
- I/O
- Network

CPU y memoria

Aunque la NASA utilizó el siglo pasado tecnología IBM Mainframe, los procesos Batch ejecutados en una institución financiera están muy lejos de la complejidad necesaria en la carrera espacial, son en su gran mayoría procesos de ordenación (SORT), manipulación de cadenas, generación de informes/interfases y cálculos sencillos.

Cualquier CPU es capaz de ejecutar ese tipo de instrucciones, sin embargo en una arquitectura IBM Mainframe monolítica, la CPU y la memoria deben compartirse entre todos los procesos ejecutados en la máquina (tareas de sistema operativo, transacciones CICS/IMS, procesos Batch, Base de Datos DB2, etc.).

Aislando cada paso (STEP) de un proceso Batch en un contenedor independiente y ejecutando el mismo sobre una arquitectura distribuida, es posible asignar recursos específicos (CPU y memoria) a la ejecución del mismo, evitando cualquier tipo de contención de recursos (WAIT).

Este tipo de arquitecturas permiten un crecimiento horizontal transparente y casi infinito, añadiendo más Hardware (nodos) a los clusters Kubernetes.

I/O

De entre todos, el recurso más crítico en un proceso Batch es el I/O. Por su naturaleza el Batch Mainframe suele estar formado por procesos secuenciales de escasa complejidad, pero muy intensivos en I/O.

En una arquitectura distribuida es posible paralelizar este tipo de procesos (i.e. Spark) y por tanto reducir el elapsed time de los mismos de manera transparente al programador de aplicaciones. El proceso principal se divide en distintos procesos de ejecución que acceden y procesan los datos de manera independiente (shared-nothing vs shared-everything architecture)

Los procesos COBOL PL/I tradicionales con acceso a ficheros (lectura/escritura) deben seguir siendo ejecutados de manera secuencial, el rendimiento de los mismos puede mejorarse de manera drástica utilizando dispositivos de almacenamiento de última generación que prescinden

de discos rotacionales.

Network

En el caso del Mainframe IBM el uso de recursos de red (Network) es irrelevante, al ejecutarse todos los componentes del proceso en el mismo Hardware.

Esta diferencia es especialmente importante en el acceso a los servidores de Base de Datos. La utilización de SQL estático junto con la ejecución de los procesos y el servidor de Base de Datos un una misma instancia del Sistema Operativo hace que sea habitual encontrar procesos de lectura masiva de datos (OPEN CURSOR) para realizar posteriormente cálculos sencillos en el programa de aplicación COBOL o PL/I (SUM, AVG, etc.).

En una arquitectura distribuida esos accesos se realizan mediante SQL dinámico y suponen un salto de red (desde el contenedor de aplicación hasta el servidor de Base de Datos) haciendo el proceso ineficiente. La solución pasa por resolver esos cálculos en el servidor de Base de Datos, recuperar la información en bloques de múltiples filas e incrementar el paralelismo del proceso.

Solución propuesta

La estrategia propuesta para el procesamiento Batch se basa en un enfoque dual, que permite atacar el “flow” de nuevos proyectos y la migración del “stock” de procesos construidos sobre tecnología Mainframe a una misma arquitectura técnica Open basada en Kubernetes.

El cliente tiene la posibilidad de;

- Migrar la funcionalidad existente, compilando los programas COBOL PL/I sobre una plataforma Linux
- Construir nueva funcionalidad sobre una arquitectura distribuida Spark con acceso a las principales fuentes de datos del Mainframe (DB2 y ficheros secuenciales)

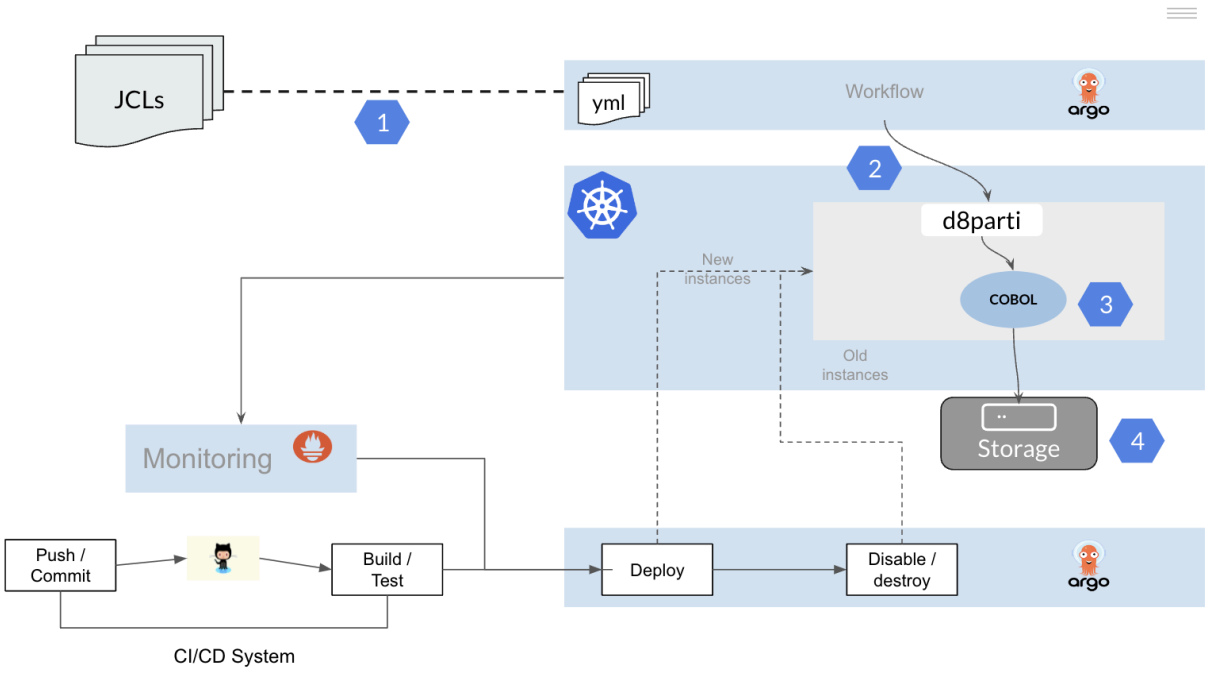
Las dos estrategias son compatibles, pudiendo existir procesos mixtos que combinen programas COBOL y Spark en distintos pasos de un mismo JOB (el STEP1 ejecuta un programa COBOL y el STEP2 un programa Spark)

Migración del stock

Para la migración de los procesos construidos sobre tecnología Mainframe es necesario replicar la funcionalidad descrita anteriormente sobre un clúster Kubernetes.

Es necesario por tanto;

1. Convertir los JCLs (JOBs) a una herramienta o framework que permita la ejecución de workflows sobre una plataforma Kubernetes
2. Replicar las funcionalidad del JES para permitir el scheduling y ejecución de los programas COBOL PL/I sobre el cluster Kubernetes
3. Recompilar los programas de aplicación
4. Permitir el acceso a los datos (ficheros y Bases de Datos)



Construcción nueva funcionalidad

Para la construcción de nueva funcionalidad Batch se propone usar un modelo de arquitectura de proceso distribuida basada en Apache Spark.

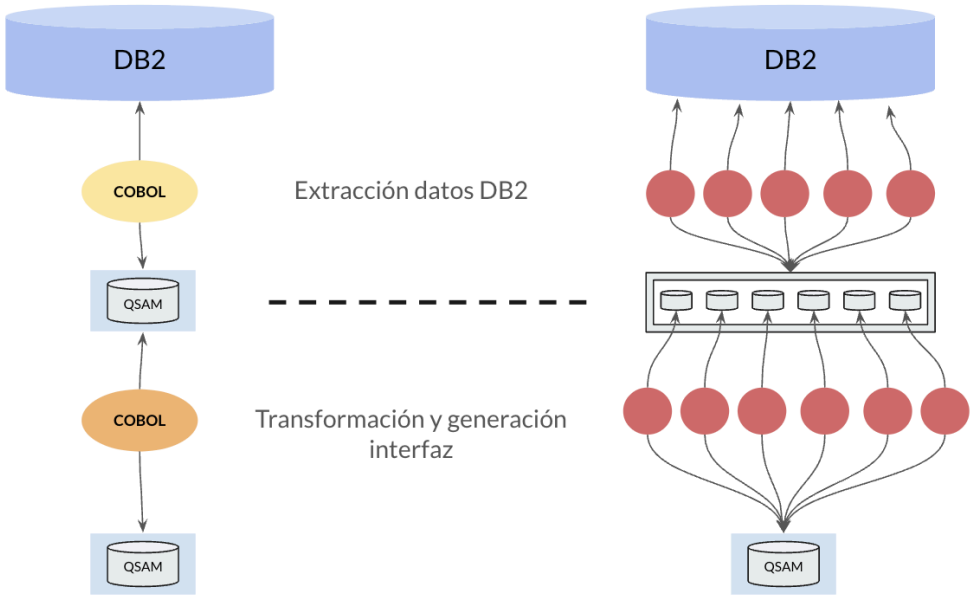
Los procesos Spark comparten con los procesos COBOL la arquitectura técnica definida anteriormente;

- La herramienta para la definición de workflows-Argo (pueden crearse jobs que mezclan pasos en ambas tecnologías)
- La plataforma Kubernetes
- Los herramientas de monitorización y logging
- El pipeline CI/CD para el despliegue de contenedores en la plataforma
- El acceso a las Bases de Datos (DB2 / Oracle) y ficheros
- Etc.

Spark permite operar con múltiples “Data Sources”, entre ellos jdbc, que puede utilizarse para acceder de manera sencilla al DB2 de la plataforma IBM Mainframe.

Los ficheros del mainframe pueden ser accedidos y transformados a distintos formatos (text, parquet, csv, JSON, etc).

Ejemplo de conversión de un proceso secuencial a un proceso distribuido spark



(1) A diferencia de proceso Mainframe COBOL, el proceso de lectura de la base de datos DB2 se puede paralelizar, por ejemplo levantando un pod Spark (executor) por cada una de las particiones de la tabla (Tablespaces particionados)

(2) La información recuperada se carga en la memoria del cluster Kubernetes o puede escribirse en un almacenamiento intermedio tipo S3 (almacenamiento objetos) en un formato columnar (parquet) manteniendo la estructura y nombres de la tabla DB2 original

(3) El proceso de transformación de la información también se puede paralelizar, el resultado final de cada Spark executor se consolidada y escribe en un fichero de salida