Visualización, modelado y monitoreo de datos de petróleo y gas con tecnologías *open source*

Por ***Julián José Benitez, Manuel Brito*** y ***Marilyn Angélica Vargas*** (Remasa)

Las tecnologías open source impulsan una gestión de datos más accesible, transparente y eficiente en el petróleo y gas. Este trabajo presenta una plataforma integral basada en código abierto para visualizar, modelar y monitorear datos críticos. Una solución flexible y escalable que ayuda a optimizar costos y mejorar la competitividad del sector.

Este trabajo fue seleccionado en las 3º Jornadas de Revolución Digital para Petróleo y Gas.

Planteo del problema

La industria del oil & gas enfrenta desafíos significativos en términos de reducción de costos y mejora en la eficiencia operativa. Con la creciente presión por ser más competitivos en un mercado exigente, las empresas necesitan soluciones innovadoras que permitan una gestión eficiente e integral de los datos críticos de perforación, reservorios y producción.

Actualmente, muchas empresas dependen de softwares y plataformas independientes para la gestión y análisis de datos, lo que limita la accesibilidad, escalabilidad y flexibilidad de las soluciones implementadas. Esta limitación puede generar una dependencia de proveedores específicos y, en consecuencia, aumentar los costos de operación.

Se presentará un caso que parte de la necesidad de visualizar, en tiempo real, las curvas de perforación con una conexión de protocolo WITSML. Esta visualización permite configurar alarmas inteligentes que facilitan la identificación de problemas tanto para el operador como para el ingeniero de perforación. En segundo lugar, se realizará un análisis histórico de pozos adyacentes previamente perforados a través de los datos históricos de los sensores, la densidad del lodo y los surveys (en formatos CSV, xlsx, .dev, etc.) a fin de obtener métricas que optimicen la perforación en pozos futuros. Adicionalmente, se podrán realizar comparaciones de los diversos parámetros de perforación y reservorios (tanto sensores como perfiles) contra el pozo que se está perforando en tiempo real, creando alarmas para optimizar el proceso. Por último, se creará una infraestructura para el desarrollo de modelos de aprendizaje automático (ML) que permite la identificación de riesgos a la operación.

El trabajo propone el uso de tecnologías de código abierto para la visualización y modelado de datos en el sector del oil & gas. Las tecnologías incluyen Dremio, Trino, Apache Iceberg, Grafana y Git, que permiten una integración eficiente y una gestión de datos transparente y escalable.

El uso de datos es fundamental en esta propuesta, ya que se busca desarrollar una plataforma que permita la visualización, documentación y disponibilidad de diversos datos de perforación, reservorios y producción. La solución puede ser implementada tanto on-premise como en la nube, adaptándose a las necesidades específicas de cada empresa.

Desarrollo Técnico del Trabajo

El primer paso fue definir cuáles serían las características claves para la solución. Las mismas fueron:

- Infraestructura flexible
  La industria del Oil & Gas presenta necesidades cambiantes y dinámicas por lo que la infraestructura debe ser flexible para poder adaptarse a cada proyecto y cada nuevo caso de uso, sea en cloud u on-premise. Es imperativo que cada adaptación requiera la menor cantidad de ajustes y cambios posibles.
- Soporte en tiempo real
  En el mundo del Oil & Gas, el procesamiento y monitoreo en real-time es necesario para monitorear y reducir riesgos en las operaciones y la identificación rápida de problemas.
- Integraciones
  El monitoreo, optimización y reducción de riesgos en las operaciones requiere de un análisis avanzado y de la integración de diversas fuentes de datos tanto actuales como históricas. Ya sean bases de datos propietarias, históricas, personalizadas o estándar, toda información es relevante para mejorar los procesos por lo que se debe disponer de la mayor cantidad de conectores posibles para importar esos datos.
- Accesibilidad de datos
  La información existente debe ser fácilmente descubierta, catalogada, compartida y comentada, para así enriquecerse a través de su uso.
- Seguridad y permisos
  El sistema debe permitir la configuración de accesos según el rol de los diferentes usuarios.
- Acceso programático a los datos
  Es fundamental que los datos y la metadata sean accesibles mediante SQL, Python u otros lenguajes de programación. Además, se debe seguir las mejores prácticas de desarrollo de software para garantizar la mantenibilidad y el versionado de los procesos.
- Soporte a casos de uso de ML
  Es necesario definir flujos de trabajo para el entrenamiento, despliegue y monitoreo de modelos de ML.
- Accionabilidad de los datos
  Los datos y visualizaciones son necesarios para facilitar la toma de decisiones.

Una vez identificadas las características clave a tener en cuenta, se determinó la combinación de tecnologías y flujos de trabajo que permitan cumplir con los requerimientos del caso de uso. Se detalla a continuación la implementación concreta de cada una de las características requeridas y cómo esa tecnología impacta en las características clave deseadas.

- **Infraestructura Flexible**

Tener una infraestructura flexible requiere que el conjunto de tecnologías utilizadas sea open source, que facilita correr el código on-premise, sin depender de un vendor específico. En la actualidad hay muchas soluciones open source validadas por la industria.

Por ejemplo S3 (Object Storage), originalmente una API diseñada por AWS para guardar archivos de manera escalable, ahora es un estándar con diversas implementaciones open source y ofrecida por diferentes cloud vendors. En el caso expuesto (on premise) se usó MiniO, pero la flexibilidad de la solución permite el cambio de proveedores sin mayores dificultades. Es importante destacar el bajo costo que tiene almacenar archivos en este sistema y su flexibilidad para tener accesos más veloces de ser requerido (con diferencia de costo).

Adicionalmente, en el desarrollo de la infraestructura se usó contenedores y Docker como orquestador. Docker habilita el ecosistema de tecnologías y soluciones open-

source, y es un estándar para la flexibilidad y adaptabilidad de soluciones de alta complejidad.

Trino, es un motor de queries distribuidas que unifica todas las bases de datos que se le conecte y las expone con un único catálogo e interfaz SQL. Es flexible, otorgando herramientas para desarrollar conectores propios en caso que el tipo de base de datos específica no tenga un conector preexistente. Es el motor detrás de AWS (Amazon Web Services) Athena, por lo que se podría reemplazar por este, si se quiere basar la infraestructura en AWS.

Apache Iceberg es un formato de almacenamiento de tablas, que permite el escaneo analítico de los datos de manera eficiente. Son archivos almacenados en S3, por lo que su costo de almacenamiento es bajo. Iceberg guarda estadísticas sobre qué tiene almacenado cada archivo, facilitando a Trino tomar decisiones inteligentes y eficientes sobre la consulta a datos almacenados.

Grafana es un software de visualización BI que incluye permisos, alertas, notificaciones y la posibilidad de visualizar datos de diversas fuentes. Se incorporaron métricas de uso de la plataforma, información de los modelos disponibles y datos y metadatos de la información almacenada en Trino. Adicionalmente esta herramienta permite, incluso al usuario final, hacer cambios en la configuración al sistema.

Keycloak, es un software de federación de usuarios y control de permisos, desplegado para controlar el acceso a la plataforma. En línea con el requerimiento de flexibilidad, permite delegar el acceso de usuarios a otras plataformas, como Google o Active Directory.

Dremio es un motor de queries, similar a Trino, que fue implementado al inicio del proyecto. También provee acceso a tablas Iceberg pero es más restrictivo respecto a la conectividad y el dialecto SQL que utiliza. Por estas razones, se migró la infraestructura a Trino. Como ambos pueden interactuar con Apache Iceberg, las mismas tablas fueron expuestas en ambos motores hasta que la migración fue completada. Esto destaca los beneficios de utilizar un formato de tabla abierta, ya que permite migrar componentes centrales como un motor de consultas sin grandes complicaciones.

- **Soporte en tiempo real**

Como Apache Iceberg es un formato de archivo, todo motor de procesamiento que sepa interactuar con este formato puede coexistir con Trino. Por lo tanto si el negocio requiere la incorporación de una fuente de datos en tiempo real, si esta es compatible con el formato Iceberg, ambas fuentes podrán coexistir. Cabe mencionar que este formato está optimizado para casos de uso analítico, como dashboards.

Para casos de uso con requerimientos específicos de real-time o muy poca latencia, se recomienda incorporar Apache Flink, un motor especializado en procesamiento de datos en tiempo real compatible con Apache Iceberg

- **Integraciones**

Dagster es un orquestador de funciones de Python que permite programar la ejecución de funciones, actualización de tablas y cualquier otro proceso que podamos expresar en Python.

Se implementó un proceso de automatización de lectura de archivos .LAS: los datos fueron cargados a un directorio predeterminado (S3 o GDrive), donde fueron procesados en Python por Dagster para cargarse a Trino. Finalmente, los datos fueron visualizados en Grafana.

Siempre que un documento pueda ser leído en Python (u otro cliente soportado por Trino o Iceberg) y su formato sea estable, puede ser incorporado al sistema y su procesamiento automatizado.

- **Accesibilidad de datos**

Grafana es utilizada para visualizar y explorar los datos disponibles a través de dashboards, comentarios, links y alarmas. Adicionalmente, la plataforma permite la personalización de gráficos y de fuentes de datos a través del desarrollo de plugins.

- **Seguridad y permisos**

Keycloak, a través de OIDC, se usa para definir permisos basados en roles para cada usuario que luego son interpretados por Grafana, Trino y Minio. También soporta manejo de usuarios tanto externos como internos a Keycloak.

La configuración de Keycloak se adapta a las restricciones de cada plataforma: Trino soporta permisos a nivel de tabla y Minio a nivel de archivo y directorio.

- **Acceso programático a los datos**

Trino expone los datos principalmente en SQL estándar, sin embargo provee una API REST, cliente en Python, y una gran lista de clientes disponibles.

Apache Iceberg si bien tiene una menor cantidad de conectores disponibles, puede ser utilizado a través de Trino para integrar todos los clientes de este último. Los conectores directos de Iceberg sirven para situaciones más especializadas, donde la performance sea crítica o se desee utilizar alguna característica intrínseca de Iceberg no soportada por Trino.

Dagster, al ser una librería de Python (además de un orquestador de datos) permite definir todas las transformaciones y carga de datos y realizar las integraciones personalizadas en código Python. Más aún, brinda acceso al ecosistema de librerías de Python y permite la integración con Git.

A través de Git se realiza el versionado, revisión, colaboración, estandarización y reutilización del código.

Utilizar Python también brinda acceso a Pytest para testear los códigos y evitar regresiones y errores.

Otro beneficio de Dagster y Git es que el mismo código que corre en la plataforma puede ejecutarse en el entorno local de un analista. Esto permite iterar y corregir localmente el código, acelerando el ciclo de desarrollo y brindando un entorno de prueba para el analista.

- **Soporte a casos de uso de Machine Learning (ML)**

MLFlow es una librería de Python para la gestión de modelos de ML y sirve para su estandarización. Se utiliza para hacer el seguimiento de experimentos, modelos, monitorear su performance y gestionar su despliegue a través de Dagster.

Dagster se utiliza para gestionar el entrenamiento distribuido de los modelos y su posterior ejecución. La plataforma se integra a MLFlow para gestionar los despliegues de modelos. Si el modelo lo soporta, y así se

lo configura, el entrenamiento y/o ejecución puede realizarse en máquinas con procesamiento dedicado (GPU y/o CPU).

- **Accionabilidad de los datos**

Grafana permite la construcción de dashboards interactivos y ofrece la posibilidad de marcar situaciones de interés para su posterior análisis. Adicionalmente, soporta el envío de alarmas y notificaciones para que los usuarios puedan accionar sobre ellas de ser necesario.

Trino administra la base de datos de eventos que usará Grafana para otorgar interactividad y colaboración a la investigación de situaciones.

Implementación de Workflow

El desarrollo del trabajo se llevó a cabo en varias etapas, siguiendo una metodología ágil para garantizar la iteración rápida y la adaptación continua a los requisitos cambiantes. Las etapas principales incluyeron:

Recolección y Almacenamiento de Datos: Se recolectaron datos en tiempo real de los sensores de perforación utilizando el protocolo WITSML e información histórica de pozos adyacentes previamente perforados, incluyendo datos de sensores, densidad del lodo y surveys en diversos formatos (CSV, Excel, .dev,...).

Procesamiento y Normalización de Datos: Se usó Python para la integración y preparación de datos previamente recolectados. Luego se usó el cliente Python de Trino para cargar los datos en tablas de Iceberg estandarizadas y archivadas en S3.

Análisis en Tiempo Real, Histórico y Comparativo: Las consultas distribuidas y el análisis sobre datos históricos se realizaron en Trino, permitiendo la comparación de parámetros de perforaciones y de datos de reservorios de pozos previos con los datos en tiempo real del pozo actual. Grafana se usó para visualizar las curvas de los sensores en real-time y facilitar la comparación de performance con pozos adyacentes. Además, la plataforma permitió, mediante el desarrollo de plugins, elaborar gráficos tanto estándares de la industria como aquellos específicos requeridos por el usuario final.

Por último, la plataforma permitió detectar tendencias a través de varias técnicas analíticas (gráficas y estadísticas) y se usó para inspeccionar, descubrir y comentar sobre los datos ingeridos por el sistema.

Desarrollo de Modelos de ML: Se desarrolló una infraestructura apoyada en Git para la gestión de código y para la colaboración en el desarrollo de modelos de ML. Los modelos se entrenaron utilizando TensforFlow y Scikit-learn con el fin de identificar riesgos durante la perforación tales como surgencias, asentamientos, situaciones de pack off y otros. Para la gestión del ciclo de vida de los modelos de ML se utilizó MLFlow, que facilita el seguimiento de los experimentos, la reproducción de los resultados y el despliegue de los modelos.

Implementación de Alarmas Inteligentes: Con el conocimiento de dominio obtenido anteriormente, los datos relevantes identificados y las métricas definidas, se entrenaron modelos para la detección de anomalías que generaron alarmas en tiempo real. Estas alarmas fueron integradas en los paneles de Grafana para una fácil identificación y acción por parte del equipo de perforación.

Resultados obtenidos

Desempeño de la Plataforma en cualidades:

El enfoque en la flexibilidad y adaptabilidad del sistema, sumado a servicios open-source, permitió la realización de cambios importantes, como la migración de Dremio a Trino sin downtime ya que ambos son compatibles con Iceberg. Esta compatibilidad permitió migrar de un sistema a otro sin pérdida de funcionalidad mientras ambos coexistían.

Adicionalmente, la integración de datos de fuentes diversas posibilitó el análisis y comparación de datos previamente almacenados en plataformas incompatibles.

La integración, a su vez, agilizó el pipeline de ML, aumentando la velocidad de incorporación de datos, su procesamiento y, finalmente, su entrenamiento distribuido.

Por último, en vez de requerir de varios programas para tener una visión completa de la operación (perforación, reservorios, etc.) se logró disponer de todos los datos y visualizaciones en un sólo sistema que integra todas las métricas, alarmas inteligentes y documentación.

Desempeño de la Plataforma en números:

Basándonos en pruebas en diversos estados de carga, podemos observar un máximo de 1 segundo de latencia entre que se procesa el dato y se dispone en Grafana.

Finalmente, la velocidad para dibujar los gráficos es negligible al seguir las mejores prácticas de diseño de queries de SQL dado que la agregación de datos se realiza en Trino, que se especializa en el procesamiento de datos.

Conclusiones

Resumen de los Hallazgos:

A través de tecnologías y servicios open source se logró integrar toda la información relevante de los distintos sistemas independientes en una plataforma centralizada transparente, escalable y accesible programáticamente en todo el ciclo de vida del dato. A través de funcionalidades de colaboración se agilizó el desarrollo de alarmas inteligentes que, integradas con los paneles de control, permiten la mejora en la velocidad de toma de decisiones en las operaciones. Adicionalmente, la adopción de tecnologías open-source disminuye los costos asociados a las licencias de softwares propietarios, permitiendo la reinversión en otras áreas críticas del negocio.

Recomendaciones para el Futuro:

La combinación entre la madurez del ecosistema open-source y la tendencia hacia la estandarización de los datos (OSDU) podrían marcar una tendencia que se dirige a soluciones enteramente abiertas, interoperables y transparentes que aumentan la flexibilidad, agilidad y eficiencia de las operaciones.

En base a lo expuesto, la adopción de datos estandarizados OSDU podría ser un próximo paso de implementación en la plataforma desarrollada. Esta acción facilitaría la posterior integración con otros sistemas que ya hayan adoptado OSDU.