SAIN4

Sistemas Avanzados de eficiencia productiva para la Industria 4.0

PROGRAMA: PROYECTOS DE I+D EN COLABORACIÓN

ACTUACIÓN: IMDECA-Proyectos de I+D en colaboración

Entregable E2.3 (E2.3- Publicable)

Informe sobre la infraestructura Big Data Analytics y Captura de Datos

Perteneciente al paquete de trabajo: PT2

Participante responsable: ITI

Mes estimado de entrega: Mes 30

RESUMEN

SAIN4 es un proyecto financiado con el Instituto Valenciano de Competitividad Empresarial (IVACE) y la Unión Europea a través del Fondo Europeo de Desarrollo Regional (FEDER).

El presente documento tiene el objetivo de recoger los resultados de construcción de la infraestructura Big Data Analytics y Captura de Datos que permiten digitalizar los procesos productivos y servir al Sistema de Gestión Avanzada (SGA) de datos necesarios para su funcionamiento.

ABSTRACT

SAIN4 is a project funded by the Valencian Institute for Business Competitiveness (IVACE) and the European Union through the European Regional Development Fund (FEDER).

The purpose of this document is to collect the results of the construction of the Big Data Analytics and Data Capture infrastructure, which will allow the digitization of the production processes and serve the Advanced Management System (AMS) of the data necessary for its operation.

Tabla de Contenidos

1 Introducción

1.1 Objetivos del Paquete de Trabajo 2

El objetivo de este paquete de trabajo (**PT2-Infraestructura Big Data Analytics y Captura de Datos)** es el de diseñar e implementar las infraestructuras de captura, almacenamiento, procesamiento y análisis de grandes volúmenes de datos que son necesarias en el marco del proyecto SAIN4. Esta infraestructura será diseñada teniendo en cuenta el ciclo de vida del proceso productivo que integra la denominada Industria 4.0. Con dicho fin, será necesario realizar un conjunto de tareas para:

- Identificar y evaluar las características de los sistemas industriales que producirán información teniendo en cuenta sus mecanismos de comunicación
- Analizar los estándares, protocolos y arquitecturas industriales de comunicación para seleccionar los más adecuados para los objetivos del proyecto y establecer un marco común de interoperabilidad de los diferentes sistemas productores de información.
- El diseño y desarrollo de la infraestructura de captación de datos a partir de sensores, dispositivos industriales y otros sistemas informáticos, que sea adaptable a los diferentes modelos de producción presentes en la industria del mueble y metal-mecánica.
- Diseño arquitectónico y desarrollo de una infraestructura Big Data Analytics que permita realizar tareas de procesamiento paralelo de grandes volúmenes da datos.
- Integración y validación de los sistemas de captura de datos con la infraestructura Big Data Analytics para implementar posteriormente un motor de prognosis.

El resultado final del paquete de trabajo será la implementación de una infraestructura que permita el desarrollo del paquete PT3, Motor de Prognosis para la eficiencia productiva, y del paquete P4, la implementación del Sistema de Gestión Avanzada para la eficiencia productiva.

1.2 Objetivo del presente documento

El objetivo del entregable E2.3 es un informe que recoge los resultados de la construcción de la de la infraestructura Big Data Analytics y Captura de Datos que permitirán obtener datos del sistema de sensorización para servirlos al Sistema de Gestión Avanzada (SGA) una vez sean procesados por el Motor de Prognosis.

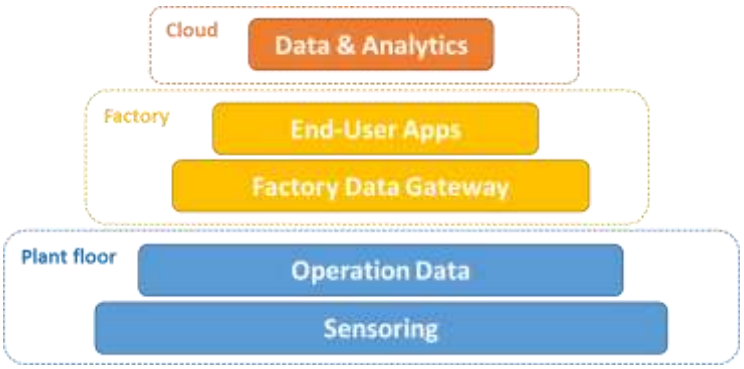


Figura 1: Arquitectura general

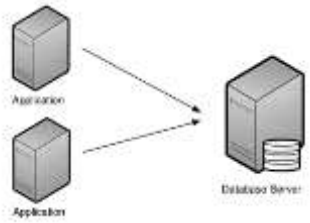
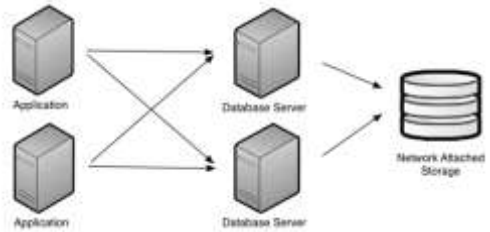
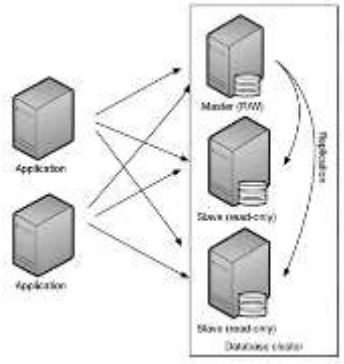
En concreto, el documento recorre cada uno de los trabajos realizados para el desarrollo, despliegue y validación de la infraestructura organizados en tres niveles de abstracción, tal y como se planteó en el entregable E2.2 y en la figura anterior:

2 Estudio comparativo de Bases de Datos NoSQL

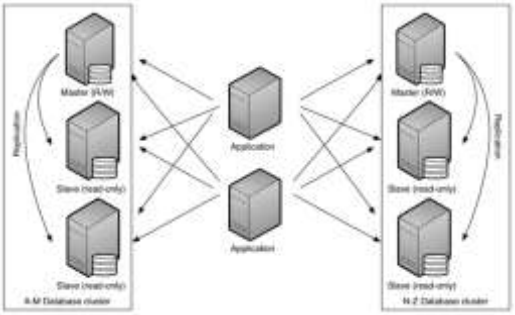
2.1 Introducción

Las bases de datos relacionales (RDBM – Relational Data Base Management) presentan problemas de escalado e interconectividad cuando aumentan las consultas en entornos Big Data ya que la complejidad en la manipulación de los datos y la interconectividad conllevan tiempos altos de respuesta. Para resolver este problema se emplean las bases de datos NoSQL las cuales permiten gestionar el volumen de datos en constante aumento y tienen una escalabilidad horizontal.

Obsérvese en los siguientes diagramas como aumenta la complejidad del sistema conforme aumentamos la escalabilidad.

Arquitectura monolítica	
Añadimos el uso de sistemas de almacenamiento en red como SAN o NAS	
Modelo Maestro - Esclavo	

Modelo Maestro – Esclavo con técnicas de particionamiento (sharding)



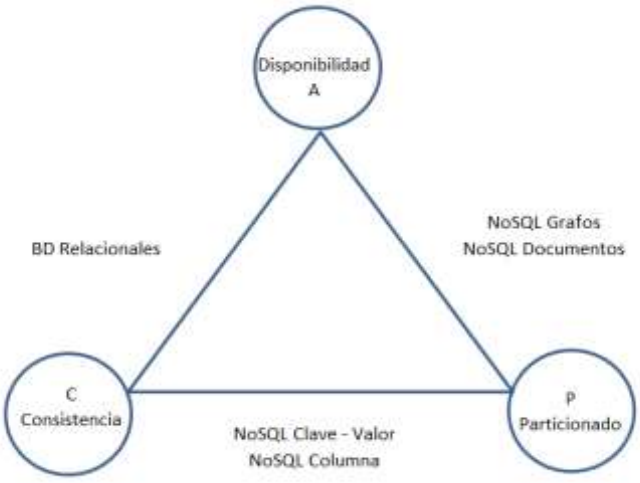
2.2 Comparativa entre bases de datos relacionales y NoSQL

Bases de datos relaciones	NoSQL
Soportan un lenguaje de consultas potente.	Lenguaje de consultas sencillo.
La estructura (schema) de los datos es fija	La estructura de los datos es variable
Cumplen las propiedades ACID	Eventualmente consistente
Soportan transacciones	No soportan transacciones

Adicionalmente si consideramos el teorema CAP (también conocido por conjetura de Brewer) cuyo enunciado indica que es imposible para un sistema de cómputo distribuido garantizar simultáneamente los tres factores siguientes:

- La consistencia. Todos los nodos ven la misma información al mismo tiempo.
- La disponibilidad. Siempre se recibe una respuesta por parte del sistema a una consulta dada.
- La tolerancia al particionado. El sistema sigue funcionando incluso si algunos nodos fallan y no pueden comunicarse entre ellos.

Gráficamente se representa de la siguiente forma:



Cómo se puede observar en el gráfico anterior, las bases de datos relacionales cumplen con dos de estos principios [disponibilidad –se puede conseguir con un clúster- y consistencia – propiedades ACID-]. Ante un fallo parcial el sistema dejará de funcionar o perderá la consistencia.

La Base de datos NoSQL Cassandra cumple con los principios (Consistencia - Particionado) que implica que para conseguir la consistencia corremos el riesgo de interrupción del servicio si un nodo falla.

ArangoDB permite al sistema centrarse en cualquiera de las propiedades CAP ya que es una base de datos NoSQL multi-modelo.

Clasificación de Bases de datos NoSQL

A. Map Reduce.

   Hadoop https://hadoop.apache.org/
   Apache Spark http://spark.apache.org/

B. Bases de datos de clave-valor

   Redis (ANSI C) http://redis.io/
   Memcached http://memcached.org/
   Amazon DynamoDB http://aws.amazon.com/dynamodb/

C. Bases de datos multicolumna

   Google BigTable (OSDI'2006) https://cloud.google.com/bigtable/
   Apache Cassandra (Facebook) http://cassandra.apache.org/
   Apache HBase (Java, sobre HDFS, como Google BigTable sobre GFS) http://hbase.apache.org/
   Apache Accumulo (NSA) https://accumulo.apache.org/

D. Bases de datos de documentos

   MongoDB (C/C++, Javascript) https://www.mongodb.org/
   Couchbase (C/C++, Erlang) http://www.couchbase.com/
   CouchDB (Erlang) http://couchdb.apache.org/
   Google Datastore https://cloud.google.com/datastore/
   Amazon DynamoDB http://aws.amazon.com/dynamodb/
   MarkLogic http://www.marklogic.com/

E. Bases de datos de grafos

   Neo4j (Java) http://neo4j.com/
   OrientDB (Java, multi-modelo) http://orientdb.com/
   Titan (Java) http://thinkaurelius.github.io/titan
   HyperGraphDB http://hypergraphdb.org/

2.3 Cassandra

Base de datos NoSQL desarrollada inicialmente por Facebook en 2007 para resolver el problema de búsqueda de mensajes en los buzones de entrada. En el 2012 se incorporó como proyecto top-level en la fundación Apache. La empresa DataStax, Inc es quien mantiene, distribuye y soporta la versión empresarial de Cassandra. También es utilizada por Netflix para personalizar la experiencia de usuario (historial, puntuaciones de películas etc) con 10 millones de transacciones por segundo. Otro caso de éxito es Instagram que utiliza una versión personalizada de Cassandra para gestionar las fotografías de los usuarios, mensajería y detección del fraude.

2.3.1 Propiedades

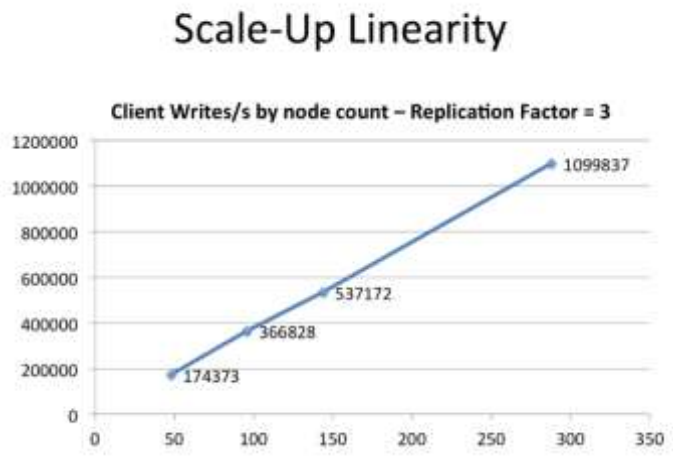
Distribuida: Puede ejecutarse en varias máquinas comportándose como un ente único.

Descentralizada:

- Todos los nodos del sistema tienen el mismo comportamiento. No hay nodos maestros ni nodos esclavo. Cualquier nodo puede tomar el papel de coordinador en una consulta.
- Es un sistema P2P que utiliza GOSSIP para mantener la sincronía entre nodos y gestionar una lista de los nodos que están vivos o muertos.
- El fallo de un nodo no provoca la indisponibilidad del sistema.

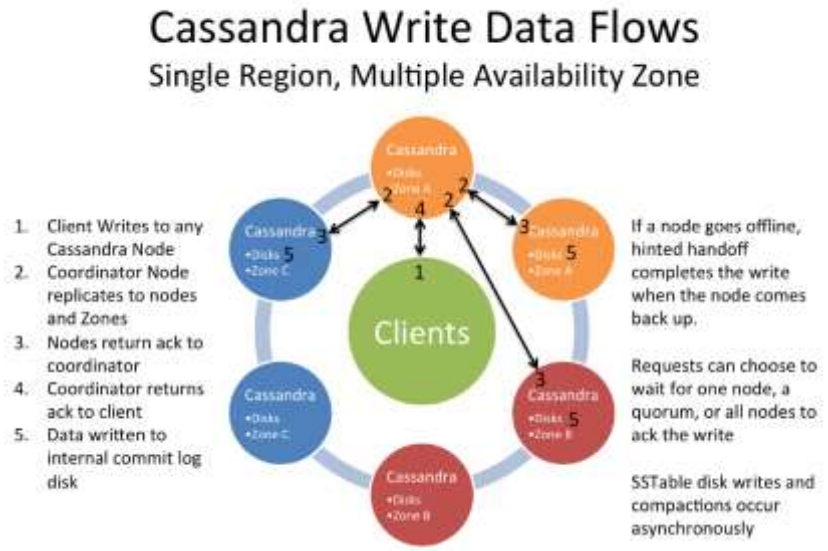
Escalabilidad elástica:

- Escalabilidad del sistema es la característica del sistema que hace que se pueda aumentar el número de peticiones atendidas con poca degradación en el rendimiento. La escalabilidad puede ser vertical u horizontal siendo la vertical las producidas a consecuencia de mejorar el hardware de un nodo mientras que la horizontal es la obtenida como resultado de añadir más nodos.



Alta disponibilidad y tolerancia a fallos

Cassandra permite una red de nodos con capacidad de redirigir las peticiones a los nodos que se encuentran operativos. También permite el reemplazo de nodos en caliente y la replicación de datos distribuidos en varios data-centers.



Consistencia

Es la capacidad de contestar las peticiones recibidas siempre con los datos vigentes. Sin embargo, en aras de mejorar el rendimiento del sistema se puede ajustar la consistencia (eventualmente consistente).

Orientado a filas

Cada registro puede tener un conjunto de columnas diferentes. Los registros tienen una clave con la que se puede acceder a los datos y que se utiliza para distribuir las filas en distintos almacenamientos de datos. A nivel físico se utiliza un sistema multidimensional de tablas hash.

2.4 ArangoDB

Hasta el 2012 conocida también como AvocadoDB. Con el mismo motor de bases de datos y utilizando el AQL (Arango Query Language) podemos realizar consultas de clave-valor, grafos, o documentos. Permite el escalado horizontal de joins, índices secundarios y transacciones ACID. Operaciones con grafos.

La base de datos permite la búsqueda de patrones, camino más corto y consulta de recorridos.

2.4.1 Propiedades

Consolidación

Base de datos multi-modelo. Al utilizar un mismo motor se minimiza el número de componentes a mantener con lo que reduciendo la complejidad del sistema obtenemos un

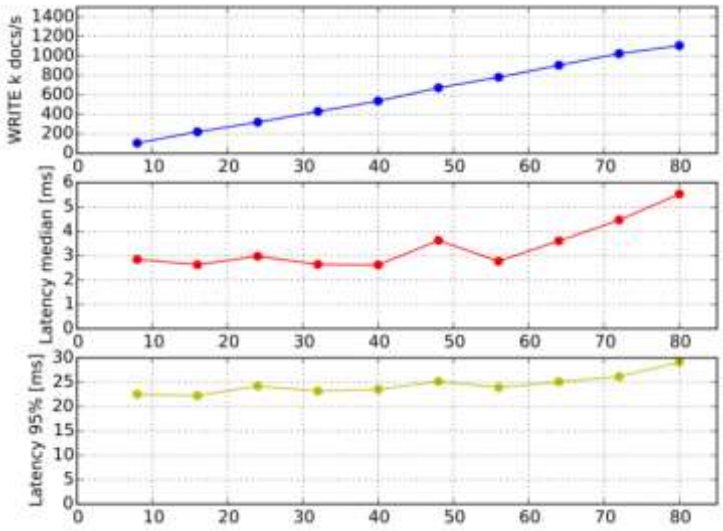
menor coste de propiedad, incremento de la flexibilidad y una mayor consolidación de las necesidades.

Escalado

Las aplicaciones crecen y evolucionan con el paso del tiempo. ArangoDB escala tanto horizontal como verticalmente. Además, si los requisitos menguan podemos reducir el back-end para ahorrar en hardware y otros requerimientos operacionales.

Durabilidad

Cuantas más veces haya que escribir en disco más bajo será el rendimiento en IO. Por otra parte, las escrituras frecuentes minimizan el riesgo de pérdida de datos si el servidor falla. ArangoDB permite al administrador especificar el grado de rendimiento y durabilidad. Esto es especialmente deseable si se trabaja con discos SSD, por ello los índices y las estructuras están diseñadas para alcanzar el máximo de rendimiento en las operaciones de escritura.



Consistencia

El back-end tiene soporte para transacciones ACID en una única instancia y operaciones atómicas en clúster lo que nos evita tener que implantar mecanismos de transacciones que guarde la consistencia de los datos entre diferentes modelos.

No puede proveer consistencia, disponibilidad y particionados de forma simultánea pero ArangoDB nos permite seleccionar las características del Teorema de CAP que más nos interesen.

Tolerancia a fallos

ArangoDB permite arquitecturas modulares y modernas con diferentes modelos de datos incluyendo la formación de clúster. Entre otros permite los siguientes tipos de replicación:

- master-master synchronous
- master-master asynchronous
- master-slave synchronous

- master-slave asynchronous

Reducción del coste de propiedad

Cada tecnología de BD necesita mantenimiento, actualizaciones, parcheo de bugs y modificaciones distribuidas por el fabricante. Cada actualización necesita ser comprobada antes de su instalación en producción. Como tenemos un único motor el número de estas operaciones se reduce al mínimo.

Comparativa de rendimiento con otros motores de BD

ArangoDB supera de forma considerable a otras alternativas del mercado. Arango consigue un pico de escritura 1730 documentos por segundo por CPU virtual mientras que algunas alternativas consiguen

- Cassandra (Google) consigue 394 documentos por segundo por CPU virtual.
- FoundationDB alcanza 750 operaciones de escritura por CPU virtual.
- Cassandra (Netflix) consigue 965 documentos por segundo por CPU virtual.
- Couchbase consigue 1375 documentos por segundo por CPU virtual.
- Aerospike 2500 documentos por segundo por CPU virtual.

3 Construcción de la infraestructura

3.1 Plant Floor

El nivel de planta o *Plant Floor* hace referencia al equipamiento y procesos que se desarrollan en el marco de la cadena de producción. Los actores principales en este nivel son por un lado el equipamiento industrial que da soporte a la producción y, por el otro lado, los operarios involucrados en las tareas de monitorización y mantenimiento.

3.1.1 Sensoring Layer

Comenzando por el ***nivel bajo de sensorización***, es obvio que se debe seleccionar las variables relevantes que pueden tener influencia en proceso, o que es necesario medir adicionalmente o en paralelo a las que regulan el funcionamiento de la propia máquina o proceso. La mayoría de procesos industriales actuales disponen de un control mediante PLC, por lo que en primera instancia la mejor opción es obtener el dato directamente de este elemento de control, teniendo la precaución de no alterar la señal de entrada o salida del PLC para no modificar el control del proceso.

Un grave inconveniente de este sistema es que en muchas ocasiones el fabricante del equipo impide el acceso al mismo, o rescinde la garantía en caso de que se haga. Afortunadamente, en los equipos de última generación es el propio fabricante quien facilita el acceso a los datos que genera la máquina, de modo que se pueden recoger directamente en algún puerto de salida. De todas formas la inmensa mayoría de equipos que están operativos en la actualidad no disponen de estas facilidades.

La alternativa en este caso es instalar ***sensores externos*** a la máquina que proporcionen los datos buscados, y gestionar las señales de estos sensores mediante un controlador que también sea independiente del proceso analizado. La selección de los sensores debe realizarse

considerando al menos tres factores: precisión de las mediciones, viabilidad de la instalación y entornos especiales de trabajo. El factor coste no se menciona pero siempre es un limitante, ya que se debe considerar el presupuesto disponible que estará íntimamente relacionado con el resultado esperado tras la instalación (mejora de la eficiencia del proceso).

La ***comunicación*** entre los sensores externos y el controlador se realiza mediante el protocolo de red EtherCAT, el más rápido actualmente disponible, con tiempos de actualización menores de 100 microsegundos.

El controlador independiente que gestiona todos los datos que se obtienen de los sensores, y en su caso de las señales de E/S de los autómatas de los equipos, debe alimentar una base de datos histórica a corto plazo que almacena temporalmente toda la información del proceso. Habitualmente se utiliza el protocolo OPC UA para transferir información desde el controlador hasta la base de datos. Actualmente existen controladores con CPU integrada capaces de gestionar directamente las bases de datos más habituales, y es la opción elegida en este proyecto.

Además de los datos de los sensores se recogen datos de análisis y verificaciones manuales, muy complejas de obtener de forma automática. Estos datos se reportan directamente por el personal responsable de realizar el análisis o verificación mediante dispositivos táctiles HMI, con interfaces diseñadas a propósito en cada caso. Dado que la carga del dato no puede realizarse en tiempo real, se habilita un registro que recoge manualmente el instante de tiempo en el que se realiza la prueba.

3.1.2 Operation Data Layer

Esta capa implementa el software necesario para recopilar la información que se produce entre los dispositivos de una planta, operación u otra distribución organizativa de una fábrica. La persistencia se implementa mediante un *Data Historian*, es decir una base de datos histórica a corto plazo con los datos de todos los dispositivos bajo un esquema común. Esta base datos se complementa con un módulo para el procesamiento de datos en “bruto”. La combinación de ambos componentes implementa un *IIoT Data Historian*, especializado en el procesamiento y almacenamiento de información proveniente de redes industriales.

Una vez la información es procesada, esta es almacenada en una base de datos histórica de corto-plazo (*Short-term Historian*) que almacenará únicamente la información generada en un rango determinado, por ejemplo, un mes, en función de los recursos disponibles y las necesidades organizativas. Para almacenar esta información se ha utilizada una base de datos relacional *SQL Server*.

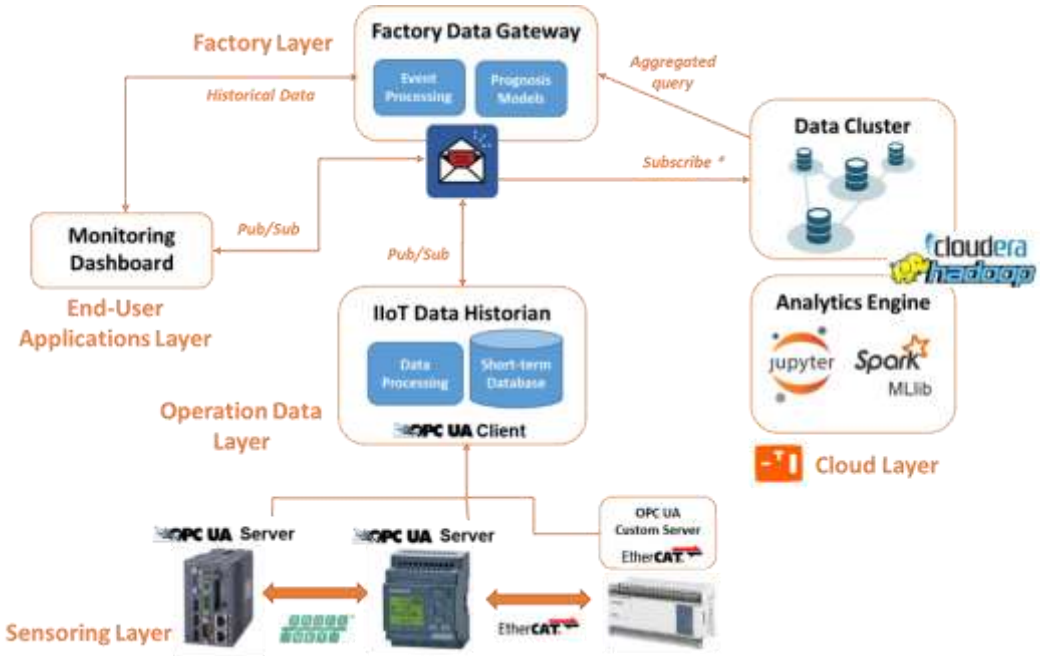


Figura 2: Arquitectura Software basada en Capas

Tal y como se puede observar en la **figura 2**, la información de este histórico debe ser almacenada también en un *Data Cluster* para tener un histórico de datos a largo plazo y poder disponer de el en función de las necesidades de los procesos a realizar.

Aunque en la **figura 2** se haya descrito que el proceso de transferencia de datos entre el *IIOT Data Historian* y el *Data Cluster* se va a realizar mediante un *public/subscriber*, el proceso final se ha desarrollado mediante un proceso ETL tal y como se muestra en la **figura 3**. Este proceso ETL se ha desarrollado en *Spark* sobre el lenguaje de programación *Python* y se encarga de realizar la conexión a la base de datos histórica de corto-plazo, extraer toda la información, filtrarla, transformarla y almacenarla en el *Data Cluster*.

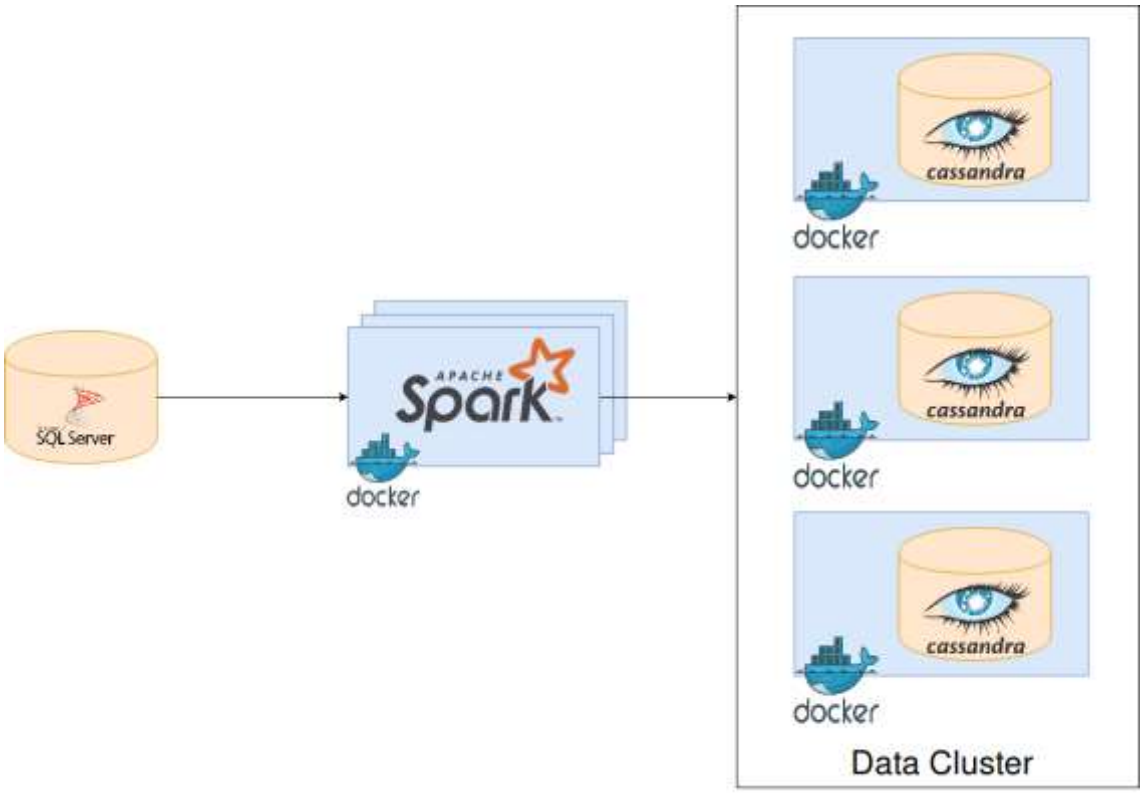


Figura 3: Proceso ETL para la copia del histórico

Para realizar el despliegue de este proceso, se ha optado por un contenedor *Docker*[1] que contiene los Scripts necesarios y una tarea programada que los ejecuta en un rango de tiempo definido por el administrador del sistema.

El beneficio de realizar esta tarea sobre *Spark* y desplegarla sobre un contenedor *Docker* es que, en caso de necesitar más potencia de cálculo, el sistema es fácilmente escalable, tal y como se muestra en la **figura 3**. En caso de querer añadir más capacidad de procesamiento al proceso ETL, se debería inicializar un nuevo contenedor *Docker* con *Spark* y añadirlo al *cluster* además de configurar este para que funcione con más de un nodo.

En la **figura 4** se puede observar un diagrama que muestra el proceso realizado en la ETL para extraer los datos de los sensores. Los pasos realizados por este proceso serían los siguientes:

- El proceso se conecta a la base de datos *SQL Server* y obtiene los datos de la tabla donde son almacenados.
- Los datos son filtrados para la máquina que se esté procesando.
- Como los datos son almacenados por filas a nivel de sensor y momento temporal, estos son pivotados para que cada registro contenga para un momento temporal los datos correspondientes a cada sensor.
- Los datos son almacenados en el *Data Cluster*.

[1] Docker

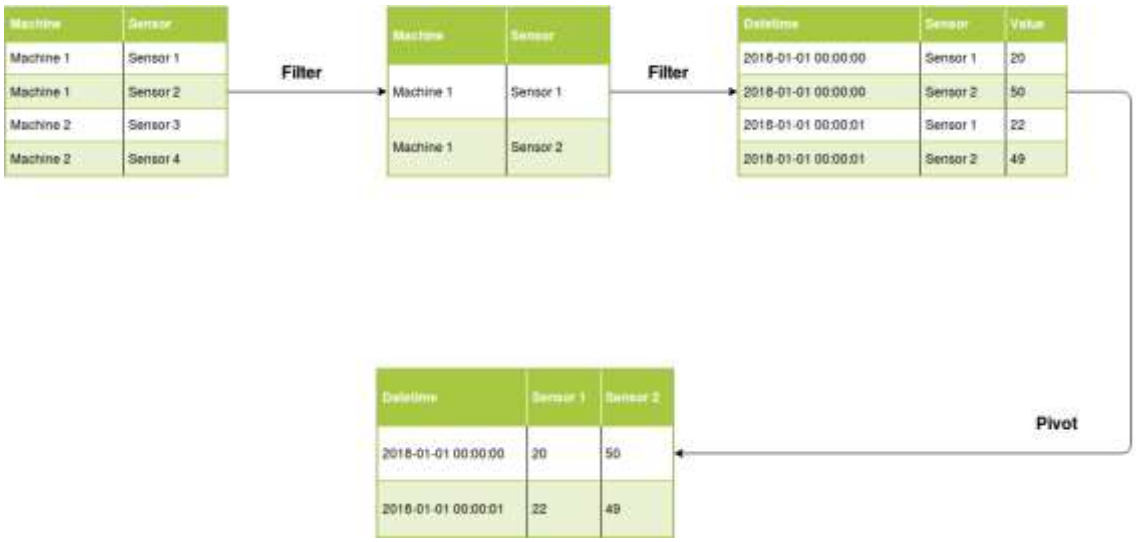


Figura 4: Transformaciones en el proceso ETL

3.2 Factory Level

Una fábrica puede dividirse en diferentes plantas o cadenas de producción que son agrupadas en el nivel *Factory* o fábrica. Por tanto, cada *Plant Floor* actuará de forma independiente y será en este nivel dónde agreguemos la información capturada mediante el *Factory Data Gateway* y facilitemos su acceso a los usuarios finales mediante aplicaciones desplegadas en la capa *End-user Apps*.

El objetivo de esta capa es aglutinar y distribuir los datos con los cuales se monitoriza el estado de la cadena de producción. Se compone fundamentalmente de un *Factory Data Gateway*, que realiza la labor de proxy para la recuperación de información de los diversos *IIoT Data Gateways* desplegados en el entorno de la fábrica. La idea es que el *Factory Data Gateway* establece un punto de acceso único a través del cual el resto de servicios acceden a los datos, sin importar si pertenecen al mismo nivel arquitectónico, capa funcional o estructura organizativa.

Como se puede observar en la **figura 2**, los datos deben ser extraídos del *Data Cluster* y ser procesados por el motor de prognosis. Los resultados obtenidos por este proceso deben ser almacenados y posteriormente se deben poder consultar de manera simple. En la **figura 5** se muestra la arquitectura elegida para poder realizar estas operaciones:

- En primer lugar, los datos son extraídos del *Data Cluster* mediante un script desarrollado en *Spark.* El script se ejecuta a través de una tarea programada que extrae los datos y los envía al motor de prognosis desarrollado en *Tensorflow*.
- El motor de prognosis realiza el estudio de estos datos y calcula los indicaroes OEE y si existe alguna anomalía en el sistema para informar de ello a los operarios. En caso de existir alguna anomalía esta será enviada al *backend* del sistema. Por su parte, los indicadores OEE son almacenados en el *Data* Cluster. Tanto el script de *Spark* como el modelo de prognosis están dentro de un contenedor *Docker*.

- El *backend* del sistema está desarrollado con el *framework Loopback* que utiliza como lenguaje de programación *Javascript.* Este *framework* recibe una petición HTTP POST en caso de existir alguna alerta, la procesa y la almacena en la base de datos *MySQL*.
- Todos los procesos anteriores se ejecutan por cada una de las máquinas que se sensorizan de manera secuencial.

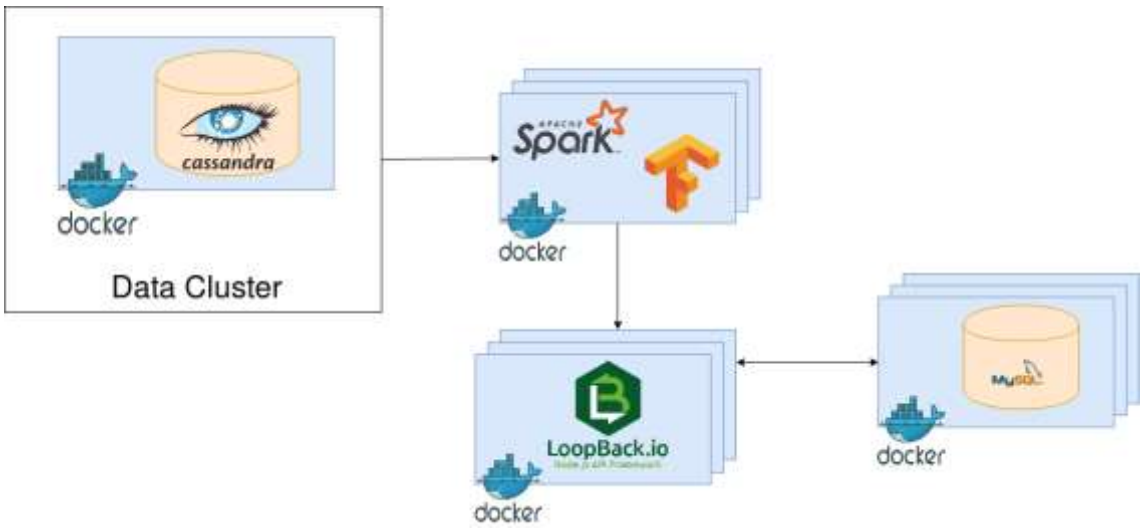


Figura 5: Arquitectura Factory Level

Como se puede observar en la **figura 5**, todas las piezas de la arquitectura están desplegadas mediante contenedores Docker, facilitando con ello el proceso de escalado, aunque actualmente solo se cuente con un nodo para cada proceso.

3.2.1 Capacidad de escalado de la infraestructura

Como se ha comentado en el punto anterior, uno de los beneficios de desarrollar la estructura mediante contenedores *Docker* es la facilidad a la hora de realizar el escalado. Existen distintas formas de poder realizar este escalado y cada pieza se puede escalar de distintas maneras.

En primer lugar, se podría realizar un escalado por proceso de modelado para cada máquina. Cada uno de estos procesos obtendría los datos del *Data Cluster*, realizara el cálculo de las anomalías y en caso de obtener alguna anomalía, enviaría esta información al *backend* para que la almacenara. Esto reduciría el tiempo de procesamiento ya que cada contenedor ejecutaría sus tareas en paralelo y enviaría la información al *backend* cuando haya finalizado, pero aumentaría la carga de trabajo que recibe el *backend*.

El esquema de este proceso puede observarse en la **figura 6**.

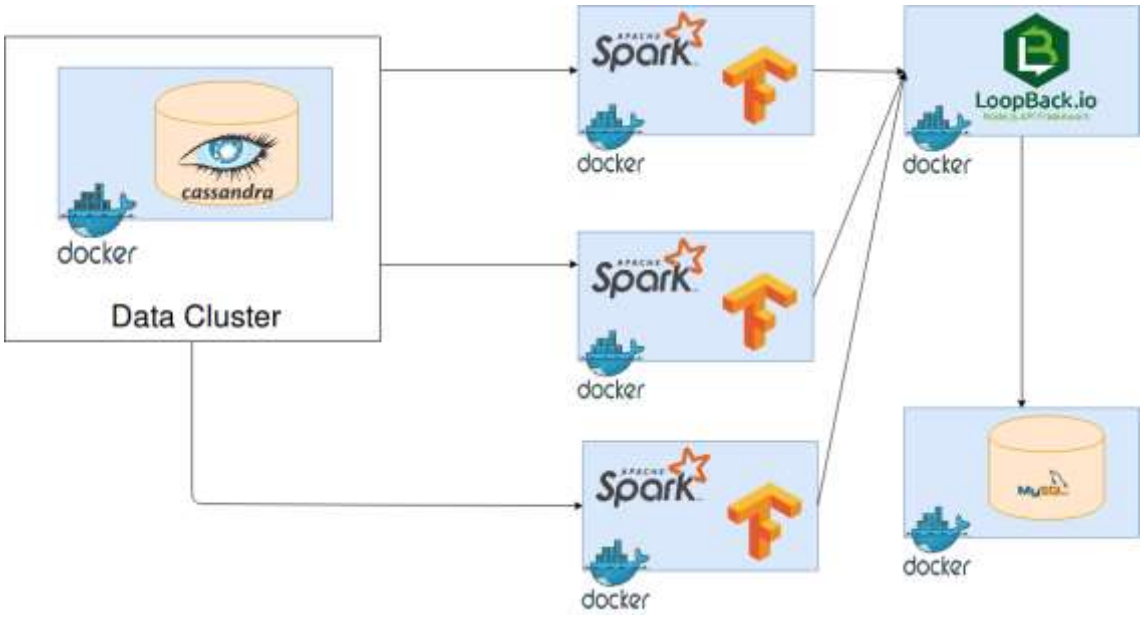


Figura 6: Arquitectura con escalado del proceso ETL y el motor de prognosis

En caso de tener que procesar muchos modelos o detectar que el *backend* tiene demasiada carga de trabajo, se podría escalar este incluyéndolo en el mismo contenedor que el proceso ETL y el motor de prognosis tal y como se muestra en la **figura 7**. Mediante esta arquitectura se reduce la carga de trabajo de cada *backend* pero se distribuyen los puntos de acceso.

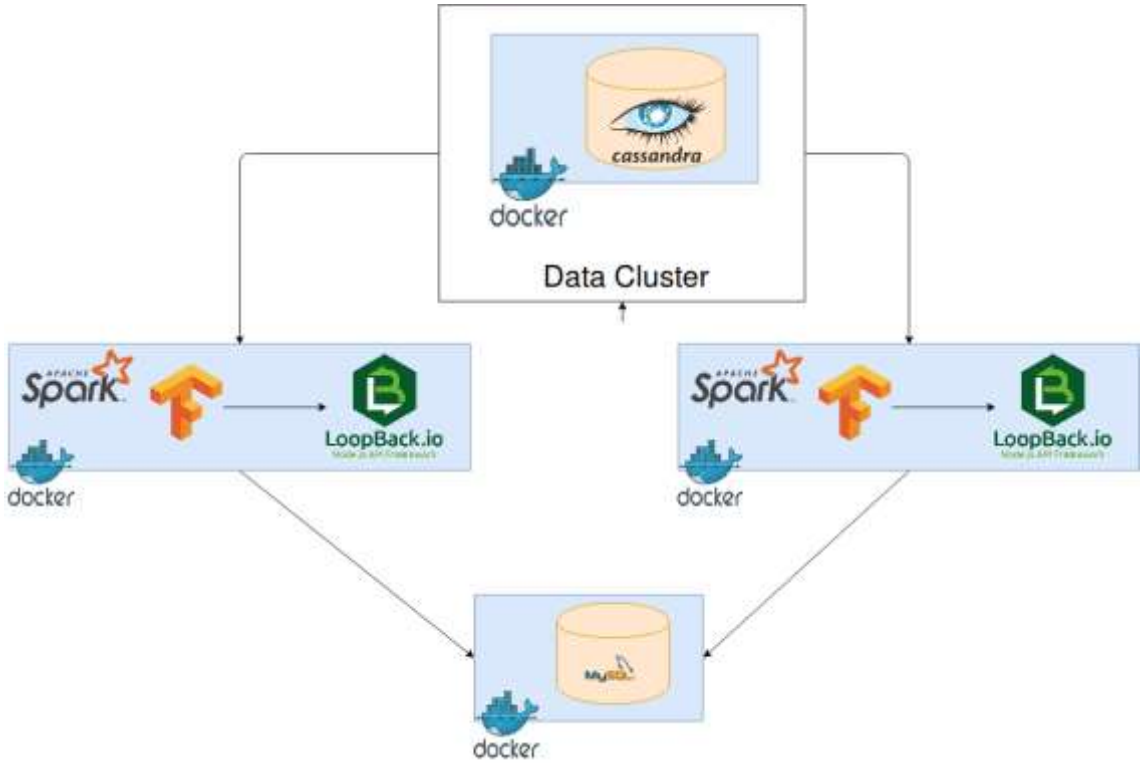


Figura 7: Arquitectura con escalado del proceso ETL y el motor de prognosis

Otro mecanismo para reducir la carga al *backend* podría consistir en replicarlo e introducir un balanceador de carga que distribuyera todas las peticiones. Esta estructura se puede observar en la **figura 8**. Con esta estructura mejoramos tanto la carga del *backend* como la paralelización de los procesos y tenemos un único punto de acceso, pero introducimos una pieza nueva al sistema.

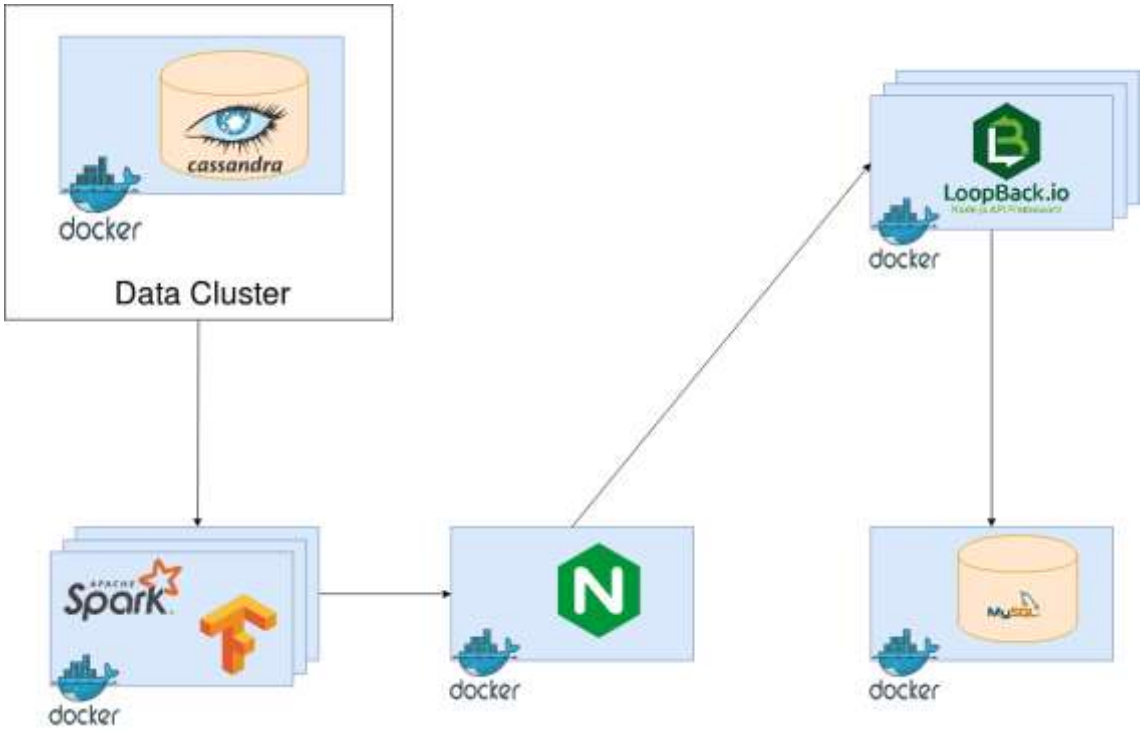


Figura 8: Arquitectura con escalado usando balanceador de carga

Otra posible solución para no incorporar un balanceador de carga pasaría por desplegar una pila de contenedores *Docker* mediante un *cluster* de Docker o *Swarm*. Este *cluster* permite ejecutar un contenedor en varias máquinas o nodos. El *swarm* está compuesto por nodos que pueden pertenecer al rol de manager o al rol de *worker*. Los nodos con el rol de manager son los únicos que pueden ejecutar comandos o que pueden autorizar a otras máquinas a entrar en el *cluster* como *workers*. Los *workers* solo proveen capacidad y no tienen la autoridad para decir a cualquier otro nodo que es lo que pueden y no pueden hacer.

Los managers de un *swarm* pueden utilizar varias estrategias para ejecutar contenedores en modo "nodo más vació", que llenan las máquinas menos utilizadas con contenedores, o "global", que asegura que cada máquina obtiene exactamente una instancia del contenedor especificado.

La ventaja de utilizar el *Swarm* es que este provee la capacidad de agregar un nodo nuevo, restaurar un nodo caído, balancear la carga de trabajo, etc. En la **figura 9** se puede observar un esquema de la representación de un *Docker Swarm*.

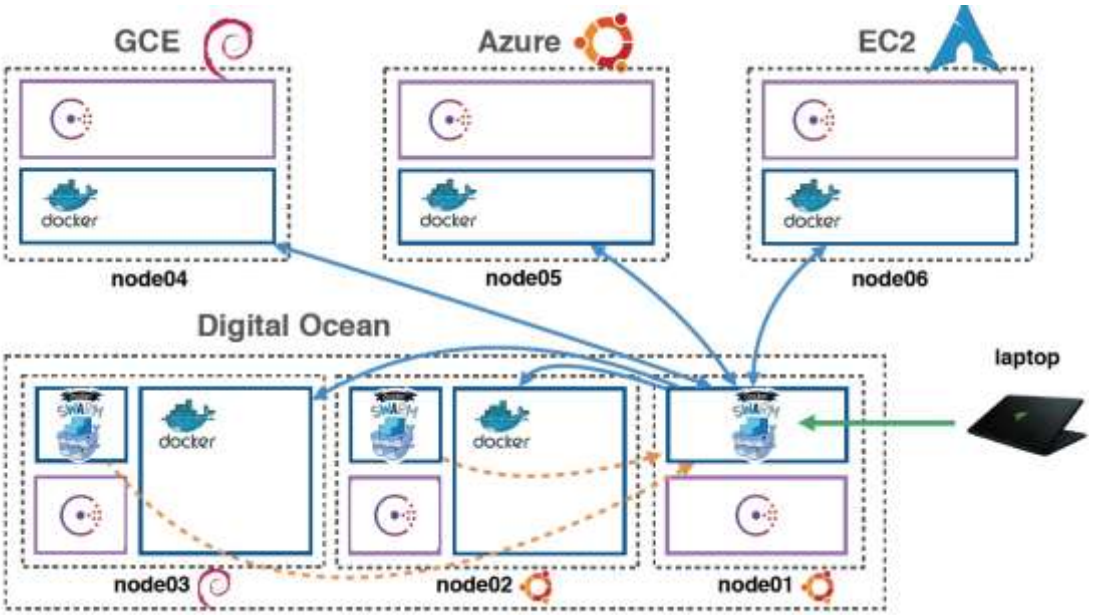


Figura 9: Docker Swarm

3.2.2 Capacidad de escalado de la base de datos

Hasta el momento solo se ha descrito el proceso de escalado de la infraestructura perteneciente a la extracción de los datos de los sensores, tratamiento de estos datos por el motor de prognosis y del *backend* que recibe las anomalías detectadas por el sistema. En este apartado se van a mostrar las distintas técnicas que permitirían escalar la base de datos en caso de que el volumen de información sea muy elevado o que esta reciba un gran número de transacciones.

Una primera opción para realizar el escalado de la base de datos consistiría en hacer *Sharding*[2]. A través de este mecanismo se puede hacer una partición de los datos dividiendo las tablas tal y como se puede observar en la **figura 10**. Tal y como se puede observar, cada contenedor de base de datos contiene la información correspondiente a un conjunto de máquinas. En caso de querer consultar la información, cada instancia del *backend* debe saber dónde consultar dicha información.

[2] Database Sharding

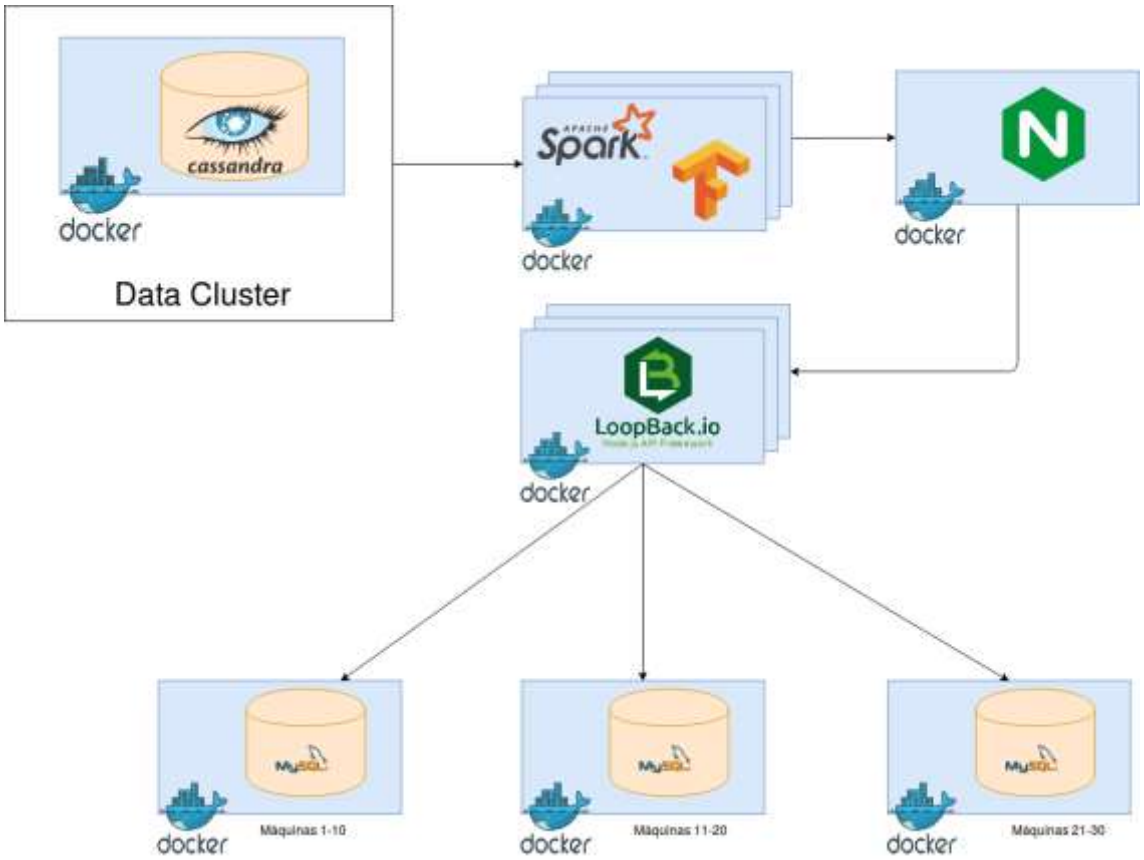


Figura 10: *Sharding* de la base de datos

Otro posible mecanismo de partición es extender la arquitectura vista en la **figura 7** y añadir a cada proceso su propia base de datos. En función de esto, todo el proceso de extracción de datos, tratamiento en el motor de prognosis, el *backend* y la base de datos se podrían distribuir por máquinas o por conjunto de máquinas.

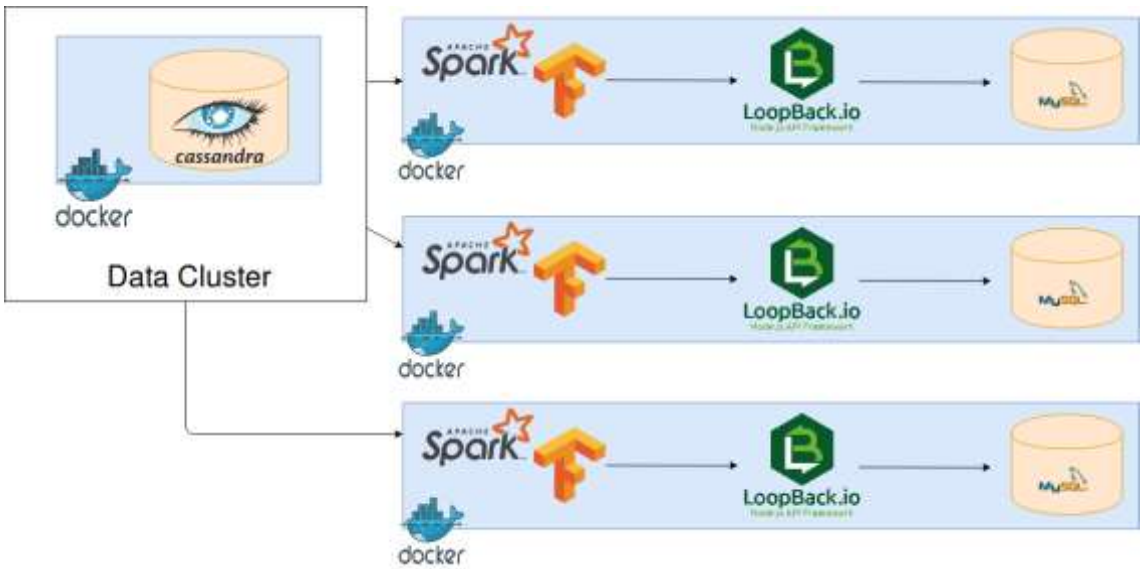


Figura 11: Distribución del proceso en distintas máquinas con su base de datos

3.3 Cloud Level

En este nivel se encuentran los servicios de persistencia y computación masiva de datos de forma externa a los sistemas informáticos de la fábrica. El objetivo es el de proveer de mecanismos de *Big Data Analytics* a la organización y agrupar la información de diferentes fábricas.

El hecho de denominarla *Cloud* hace referencia a la necesidad de escalabilidad bajo demanda que define este nivel. A efectos prácticos, este nivel puede componerse de los sistemas o servidores informáticos que posea la empresa, junto con los servicios externos que sean necesarios para soportar las necesidades de computación o almacenamiento adicionales. Esta infraestructura informática se diferencia de la presentada en los anteriores niveles por su alto nivel de sofisticación: mientras que en los niveles de planta y fábrica se utilizan PCs de carácter industrial, en este nivel es de espera un clúster o servidor avanzado.

Está capa define los componentes software para dar soporte a las necesidades de *Big Data Analytics* de la organización y a la persistencia de la información. Por la estrecha relación entre ambas tareas, se ha decidido que formen parte de una misma capa.

El *Cloud Level* está separado en dos partes, una que contiene el almacén de datos y otro que contiene el conjunto de herramientas que van a procesar y tratar estos datos. En la **figura 2** se puede observar esta estructura.

El *Data Cluster* cuenta con un conjunto de máquinas virtuales que contienen los servicios necesarios para poder almacenar la información. Cada máquina o nodo del *cluster* puede ser escalada para en caso de necesidad, aumentar la capacidad de almacenamiento.

Por otro lado, los nodos pertenecientes al proceso de *Analytics Engine* cuentan con las herramientas necesarias para realizar los procesos de ETL y de predicción sobre los datos almacenados en el *Data Cluster*.

Todas las operaciones necesarias se describen en los siguientes puntos:

- Almacenamiento de los datos que provienen de los datos generados por la planta o conjunto de plantas de la empresa.
- Transformación y procesamiento de estos datos para obtener información relevante que pueda ser mostrada a los operarios.
- Transformación y procesamiento de los datos para ser inyectados en un modelo que realice cálculos y predicciones relevantes para los operarios.
- Almacenamiento de estos datos calculados para tener persistencia y que puedan ser consultados en cualquier momento.

Dentro de todo este conjunto de operaciones, se identifican varias herramientas necesarias para poder realizarlas. Entre estas herramientas se encuentran las siguientes:

- Una base de datos NOSQL, en este caso la elegida es *Cassandra*.
- *Apache Spark*, un *framework* que permite realizar los procesos ETL y almacenar los resultados en la base de datos.
- *Scikit Learn*, que contiene un conjunto de librearías de *machine learning* para realizar los modelos indicados anteriormente.

Los nodos pertenecientes a la base de datos *Cassandra* estarían ubicados dentro del *Data Cluster*. Por otro lado, los nodos pertenecientes a *Apache Spark* y el modelado de los datos pertenecen al *Analytics Engine*. Cabe destacar que cada nodo de *Analytics Engine* contendría todas las herramientas necesarias para poder realizar las tareas descritas.

Como cada nodo se despliega sobre una máquina virtual, es posible realizar el escalado de estos. El proceso consistirá en configurar cada *Cluster* para que aceptara más de una máquina y en función de la necesidad, añadir nuevas máquinas.

4 Validación de la infraestructura Big Data Analytics

En este punto se va a validar la infraestructura Big Data Analytics. Para ello, se van a realizar un conjunto de pruebas descritas en los siguientes puntos:

- Desplegar un conjunto de nodos mediante *Docker Swarm*.
- Añadir un nuevo nodo al *Docker Swarm*.
- Provocar la caída de un nodo y comprobar que este se vuelve a levantar y se incorpora al *cluster*.
- Pruebas de estrés sobre *Cassandra*.

Mediante este conjunto de pruebas se pretende demostrar que se ha generado una infraestructura robusta, que es capaz de ser escalada sin problemas, que balancea la carga entre sus nodos y que soporta un gran tráfico de datos.

4.1 Desplegar un conjunto de nodos mediante Docker Swarm

Docker Swarm permite la configuración del despliegue de todas las instancias de los contenedores necesarios para la configuración de un *cluster*. A través de un fichero de configuración se puede definir cuantas máquinas van a conformar este *cluster*, cuantas replicas se necesitan y el mecanismo de reinicio deseado.

Para poder generar un *cluster* primer se han creado todas las máquinas virtuales necesarias. Para simplificar el problema se han generado dos, una que realizara las tareas de manager y otra que realizara las tareas de *worker*.

Una vez se han creado estas máquinas virtuales, se han unificado en un *swarm* para indicar que forman el *cluster*. Este proceso implica identificar cuál de las máquinas actuara como manager y cual como *worker*.

Una vez conectados los dos nodos entre sí, se ha creado un fichero de configuración que permite el despliegue de los contenedores y la política que va a utilizar en ese despliegue. En la **figura 12** se observa esta configuración:

```
version: "3"
services:

  loopback:
    image: loopback/sain4
    ports:
      - "3000:3000"
    deploy:
      mode: 'global'
      restart_policy:
        condition: on-failure
    networks:
      - sain4v_network

  visualizer:
      image: dockersamples/visualizer:stable
      ports:
          - "28080:8080"
      volumes:
          - "/var/run/docker.sock:/var/run/docker.sock"
      deploy:
          placement:
              constraints: [node.role == manager]
      networks:
        - sain4v_network

networks:
  sain4v_network:
```

Figura 12: Configuración del *cluster*

Como se puede observar en la imagen, se ha creado un contenedor donde se despliega la aplicación *loopback* en modo de despliegue "global". Con este método de despliegue, cada máquina virtual contendrá una instancia de *loopback*. En la **figura 13** se puede observar los resultados de desplegar sobre una única máquina virtual el *swarm*:

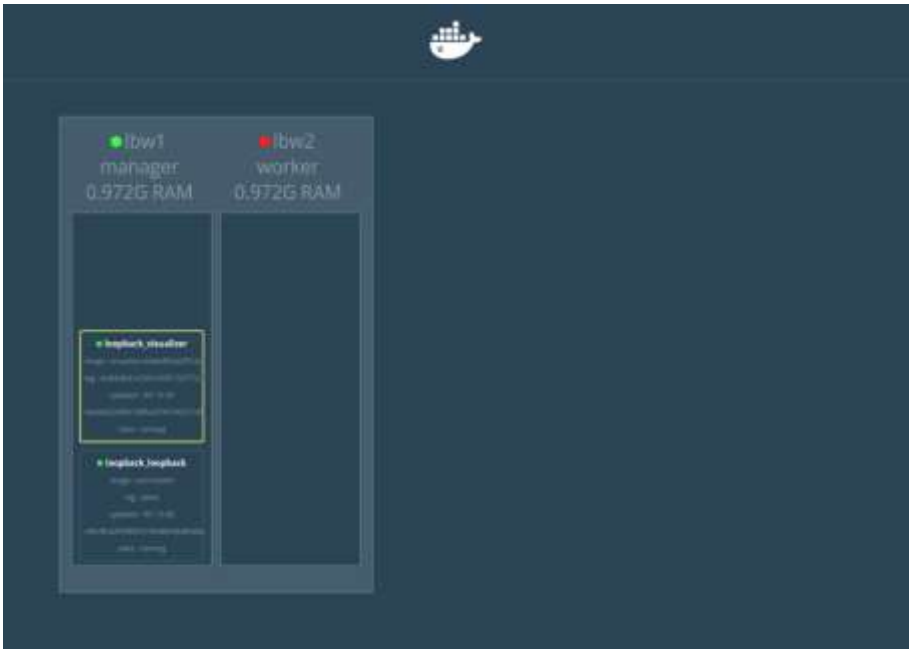


Figura 13: Despliegue del *swarm* sobre una máquina virtual

Como se observa en la imagen, el *swarm* cuenta con dos máquinas, una que está apagada y otra en ejecución. La máquina en ejecución cuenta con dos contenedores, uno que contiene la aplicación de monitorización de los contenedores y otro que es la propia aplicación del *backend*.

4.2 Añadir un nuevo nodo a Docker Swarm

Una vez generado el *cluster*, la siguiente prueba es añadir un nuevo nodo y comprobar que este se despliega correctamente. La prueba realizada para ello consiste en volver a ejecutar la máquina virtual que está apagada y comprobar que esta máquina despliega el contenedor del *backend*. Los resultados de esta prueba se pueden observar en la **figura 14**:

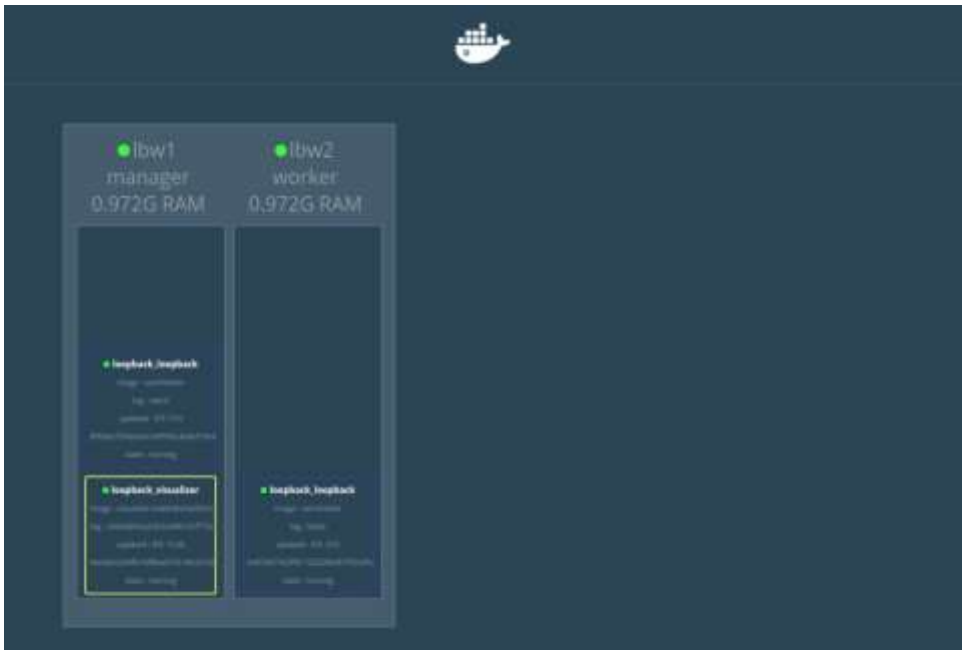


Figura 13: Añadir nodo al *swarm*

Como se puede observar en la imagen, la nueva máquina está en ejecución y contiene una instancia de la aplicación de *loopback*.

4.3 Provocar la caída de un nodo y su reincorporación al cluster

Una de las características principales que se le pide al *cluster* es que sea capaz de volver a incorporar una máquina que ha dejado de funcionar por cualquier fallo para no reducir sus prestaciones.

En este punto se ha generado un fallo en la aplicación *backend* para provocar la caída de la máquina virtual. A través de la aplicación de monitorización del *cluster* se puede observar como esta máquina deja de funcionar y como automáticamente se vuelve a ejecutar.

Como la aplicación sigue con el fallo provocado, a cada intento de ejecución se vuelve a apagar. Una vez este problema es corregido, la próxima vez que se ha vuelto a intentar ejecutar, esta se queda en ejecución y se añade al *cluster*.

4.4 Pruebas de estrés sobre Cassandra

En este punto se describen las pruebas de estrés realizadas sobre *Cassandra*. Para probar el rendimiento de un clúster de *Cassandra* es posible usar la herramienta *cassandra-stress*, se trata de una herramienta incluida con la instalación de *Cassandra* que realiza una serie de cargas sintéticas de lecturas y escrituras en el clúster de *Cassandra*.

Para realizar un test de rendimiento sobre el esquema se define un fichero en formato YAML con los datos que necesita la herramienta *cassandra-stress*, a continuación, se explica el contenido de las secciones del fichero. En primer lugar, se define el nombre del *keyspace* a testear y la consulta CQL usada para crearlo:

```
keyspace:  sensors_stress
keyspace_definition: |
   CREATE KEYSPACE sensors_stress WITH replication =
   {'class': 'SimpleStrategy', 'replication_factor': 3};
```

En segundo lugar, se define el nombre de la tabla que se va a probar junto con la consulta CQL usada para crearla:

```
table: machine
table_definition: |
   CREATE TABLE sensors_stress.machine (
   timestamp timestamp,
   sensor_temp float,
   sensor_amp float,
   ...

   PRIMARY KEY (timestamp)) WITH CLUSTERING ORDER BY (timestamp DESC)
   ...;
```

En tercer lugar, se define la distribución estadística que siguen los datos de las columnas para poder generar los datos de forma aleatoria:

```
columnspec:

- name: timestamp
  size: fixed(8)
  population: seq(1..1000000)
  cluster: fixed(1000)

- name: sensor_temp
  size: fixed(4)
  population: uniform(1..100)
  cluster: fixed(2)

- name: sensor_amp
  size: fixed(4)
  population: uniform(1..100)
  cluster: fixed(2)

...
```

En cuarto lugar, se define el *batch* usado para inserciones y consultas sobre el esquema:

```
insert:
   partitions: fixed(1)
   select: fixed(1)/1000
   batchtype: UNLOGGED
```

Por último, se definen las consultas a realizar sobre el esquema:

```
queries:
  query1:
    cql: select * from machine LIMIT 1
```

A continuación, se muestra un ejemplo de la salida del comando *cassandra-stress* con el fichero YAML definido anteriormente. Aquí se muestran datos sobre el número de operaciones por segundo, latencias y tiempo total de duración de la prueba.

```
Results:
Op rate                   :       999 op/s  [insert: 999 op/s]
Partition rate            :       999 pk/s  [insert: 999 pk/s]
Row rate                  :       999 row/s [insert: 999 row/s]
Latency mean              :       0.6 ms [insert: 0.6 ms]
Latency median            :       0.5 ms [insert: 0.5 ms]
Latency 95th percentile   :       0.7 ms [insert: 0.7 ms]
Latency 99th percentile   :       1.2 ms [insert: 1.2 ms]
Latency 99.9th percentile :       34.4 ms [insert: 34.4 ms]
Latency max               :       116.0 ms [insert: 116.0 ms]
Total partitions          :       150,000 [insert: 150,000]
Total errors              :       0 [insert: 0]
Total GC count            :       3
Total GC memory           :       1.804 GiB
Total GC time             :       0.2 seconds
Avg GC time               :       74.3 ms
StdDev GC time            :       27.3 ms
Total operation time      :       00:02:30
```

5 Validación de la infraestructura de Captura de Datos

En este punto se va a validar la infraestructura de Captura de Datos. Para ello, se van a realizar un conjunto de pruebas descritas en los siguientes puntos:

- Comunicación de los sensores
- Gestión de la base de datos a corto plazo

Mediante este conjunto de pruebas se pretende demostrar que se ha generado una infraestructura que es capaz de recoger información en tiempo real de los procesos productivos.

5.1 Pruebas de comunicación

En este punto se describen las pruebas realizadas para verificar el funcionamiento de los sensores externos e interfaz HMI para carga de datos manual. La validación se realiza utilizando cuatro tipos de sensores, una sonda de temperatura Pt100, apta para líquidos, dos sondas de temperatura y humedad ambiente, un sensor inductivo para medir la cadencia de paso de una pieza metálica con objeto de medir la velocidad de la línea y un transformador de intensidad para medir potencia eléctrica.

Excepto el sensor inductivo que da un pulso digital cuando detecta metal, los otros sensores se conectan a tarjetas analógicas de entrada del controlador NJ 100. Este controlador recoge los datos de las sondas periódicamente con la frecuencia que se establezca y los vuelca en una base de datos MySQL, configurada como muestra la siguiente tabla:

REGISTRO_1s		
ID	int	Unchecked
TRAFO1	float	Checked
TRAFO2	float	Checked
TRAFO3	float	Checked
TRAFO4	float	Checked
TRAFO5	float	Checked
TRAFO6	float	Checked
TRAFO7	float	Checked
TEMP_SECADO	real	Checked
TEMP_P_CORTINA	real	Checked
TEMP_P_RODILLO	real	Checked
TEMP_P_FONDO	real	Checked
TEMP_ZONA1	real	Checked
HR_ZONA1	real	Checked
TEMP_ZONA2	real	Checked
HR_ZONA2	real	Checked
V_IMPRIMACION	float	Checked
V_FONDO	float	Checked
V_CORTINA	float	Checked
AL_T_Z1	bit	Checked
AL_H_Z1	bit	Checked
AL_T_Z2	bit	Checked
AL_H_Z2	bit	Checked
AL_L1	bit	Checked
AL_L2	bit	Checked
AL_L3	bit	Checked
AL_L4	bit	Checked
AL_L5	bit	Checked
AL_L6	bit	Checked
AL_L7	bit	Checked
AL_T_FONDO	bit	Checked
AL_T_SECADO	bit	Checked
AL_T_RODILLO	bit	Checked
AL_T_CORTINA	bit	Checked
DATE_TIME	datetime	Checked

Además de los registros correspondientes a los valores leídos por los sensores, se han incluido un conjunto de registros que permiten indicar valores límite de algunas variables, para que el sistema lance una señal de aviso si se superan los límites establecidos. Esta funcionalidad no es estrictamente necesaria a efectos del proyecto, pero aporta un valor añadido interesante desde el punto de vista del usuario.

Seguidamente se realiza una validación de los sensores de forma individual, definiendo su modo de conexión y funcionamiento.

a) En primer lugar se verifica el procedimiento de instalación de la sonda Pt 100

b) Seguidamente se valida el sensor de Tª y HR. El sensor trabaja con la norma de bucle de intensidad 4-20 mA con 2 hilos. Dispone de un canal para cada sensor: uno para temperatura y otro para humedad.

Inicialmente se realiza un montaje preliminar con fuente de alimentación de 24V, tomando lectura de intensidad con el polímetro (comprobando que la lectura mostrada por pantalla coincide con el dato de intensidad leído). Se utiliza una resistencia de 220Ω.

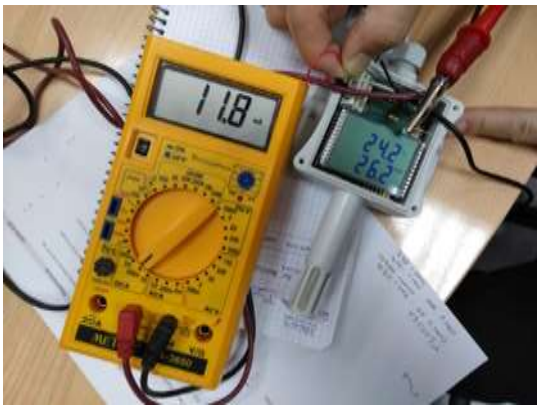
Lectura de 8’5 mA (8’5-4) x (100/16) =28’13 %.

Mostrado en pantalla 28’7 %.

Se repite el montaje para el chanel 2 (Tª). Si alimentamos +I2, -I2 no funciona. Hay que alimentar también +I1, -I1. Se puentea la alimentación desde +I1 y se mide.

Lectura de 11’8 mA ((11’8-4) x (110/16)) - 30 =23’63 ºC.

Mostrado en pantalla 24,2 ºC.

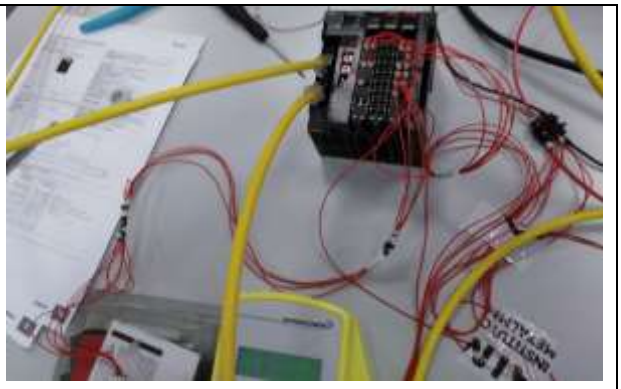


c) Revisión del funcionamiento del sensor para medir velocidad de la línea, de tipo inductivo. El amplificador se alimenta en 14-15 a 24V por medio de la masa de la cabecera del autómata y del IOV de la tarjeta digital. La señal IN de la tarjeta digital se conecta a la 7 del amplificador.

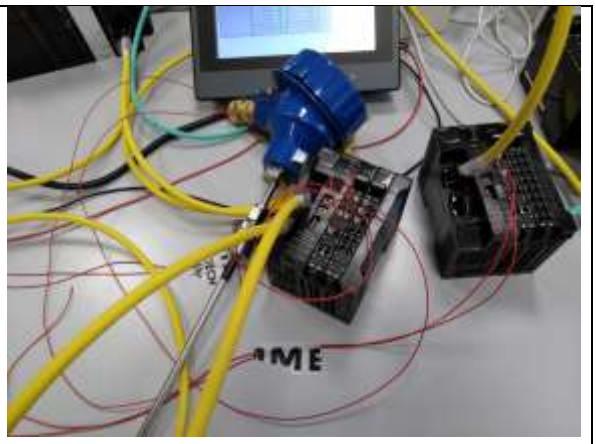
El sensor debe de estar a menos de15 mm de las placas de metal si estas son de acero. Para otro tipo de metales hay que aplicar un factor de corrección haciendo que la distancia sea menor: Acero = 1 / inox aprox. 0,7 / latón aprox. 0,5 / aluminio aprox. 0,4 / cobre aprox. 0,3.

Seguidamente se muestran algunas fotos de la instalación realizada.

Sensores de Tª y HR



Tarjeta E/S analógica 4-20 mA

	
CPU del autómata	Sonda Pt 100, cabecera remota y tarjeta E/S analógica

En las imágenes anteriores puede apreciarse los cables de red EtherCAT que comunican la CPU con las cabeceras remotas que se colocan cerca de la ubicación de cada grupo de sensores, dado que las señales de éstos no conviene que se transmitan directamente a grandes distancias (decenas de metros en el piloto que se instalará posteriormente en la empresa ROYO). Igualmente la CPU se conecta a un portátil en el que está instalada la base de datos MySQL mediante una red Ethernet IP.

5.2 Prueba de gestión de la base de datos a corto plazo

Para visualizar los datos adquiridos y depositados en un MySQL que será nuestra base de datos a corto plazo, se desarrolló una aplicación tipo SCADA que mostrase tanto en el portátil como en la pantalla HMI (dispositivo táctil industrial) las lecturas de los sensores en las diferentes zonas del proceso, incluyendo alarmas de advertencia cuando se superasen los límites preestablecidos. Algunas de las pantallas desarrolladas se muestran seguidamente.

	
Vista general del sistema SCADA en el portátil	Vista general del sistema SCADA en la HMI

	
Lecturas de los sensores para la zona de Acabado	Lecturas de los sensores para la zona de Preparación

En la página siguiente puede verse los datos adquiridos durante la prueba en los primeros 49 segundos de funcionamiento, ya que la frecuencia de adquisición se estableció en un segundo. Puede apreciarse que, en este caso, no había alarmas programadas ya que están todas a cero.

Algunos de los datos no son reales, los de velocidad concretamente, ya que no se podía simular el movimiento de una cinta transportadora.

No obstante, la prueba demostró que tanto los sensores como la electrónica intermedia y el controlador seleccionados, podían realizar la captura y gestión de datos sin ningún problema.

GENERALITAT VALENCIANA | TOTS A UNA veu
IVACE INSTITUT VALENCIÀ DE COMPETITIVITAT EMPRESARIAL
UNIÓN EUROPEA Fondo Europeo de Desarrollo Regional
Una manera de hacer Europa
"Proyecto cofinanciado por los Fondos FEDER, dentro del Programa Operativo FEDER de la Comunitat Valenciana 2014 - 2020"

ID	TRAFO1	TRAFO2	TRAFO3	TRAFO4	TRAFO5	TRAFO6	TRAFO7	TEMP_SECA	TEMP_P_CO	TEMP_P_RO	TEMP_P_FO	TEMP_ZONA	HR_ZONA1	TEMP_ZONA	HR_ZONA2	V_IMPRIMA	V_FONDO	V_CORTINA	AL_T	AL_H	AL_T	AL_H	AL_L	AL_L	AL_L	AL_L	AL_L	AL_L	AL_L	AL_T	AL_T	AL_T	AL_T	DATE_TIME
1	5,3687501	5,5250001	5,51875019	5,73125029	2,8062501	30,6312504	29,8062496	86,175	25,3125	26,5	35,9375	29,24875	36,875	30	43,225	9,66733456	6,88356161	35,6756744	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:15:05.580
2	5,3687501	5,5250001	5,51875019	5,7249999	2,81875014	30,4750004	29,6875	86,325	25,3	26,45	35,8125	29,24875	36,775	30,0375	43,225	9,82484722	6,88552666	35,402298	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:15:36.240
3	5,3499999	5,5250001	5,51250029	5,73750019	2,82500005	30,3812504	29,6000004	86,375	25,325	26,4625	35,675	29,29	37,3125	30	43,4	9,76913738	6,88552666	35,6756744	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:16:36.237
4	5,3812499	5,54375029	5,53125	5,73750019	2,8125	30,3937511	29,7124996	86,275	25,325	26,5125	35,6	29,3175	37,1125	30,0375	43,4875	9,76913738	6,88552666	35,6756744	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:17:36.273
5	5,3812499	5,55000019	5,54375029	5,76250029	2,8062501	30,5187511	29,7687511	86,325	25,35	26,5375	35,6	29,33125	37,0125	30,0375	43,55	9,7651825	6,88552666	35,5384598	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:18:36.240
6	5,3812499	5,54375029	5,5374999	5,75	2,81875014	30,46875	29,6687508	86,3625	25,4	26,525	35,6875	29,3175	36,8125	30,0625	43,4875	9,82885075	6,881598	35,402298	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:19:36.243
7	5,3499999	5,5250001	5,48750019	5,71875	2,81875014	30,4375	29,5562496	86,4375	25,4	26,5625	35,7	29,29	37,1125	30,0625	43,425	9,66733456	6,88552666	35,6756744	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:20:36.230
8	5,3375001	5,5062499	5,4937501	5,7249999	2,8125	30,7312508	29,9437504	85,85	25,3875	26,55	35,625	29,3175	37,2875	30,1	43,2625	9,83285809	6,88552666	35,6756744	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:21:36.233
9	5,36250019	5,5	5,48125029	5,7125001	2,8125	30,3937511	29,5500011	85,8875	25,4	26,6	35,6125	29,33125	37,25	30,1375	43,3875	9,84088135	6,88552666	35,6756744	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:22:36.307
10	5,375	5,55000019	5,5374999	5,75	2,8062501	30,5562496	29,71875	86,05	25,425	26,5875	35,575	29,35875	37,1375	30,1375	43,425	9,84891796	6,881598	35,5384598	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:23:36.230
11	5,35625029	5,51875019	5,5062499	5,7249999	2,79999995	30,5750008	29,7124996	86,325	25,4125	26,55	35,6625	29,3725	37,2375	30,1625	43,425	9,66733456	6,88552666	35,6756744	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:24:36.233
12	5,34375	5,51250029	5,5062499	5,7249999	2,8062501	30,4125004	29,6750011	86,325	25,4875	26,575	35,6625	29,41375	37,025	30,1625	43,4875	9,84088135	6,88552666	35,5384598	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:25:36.237
13	5,3687501	5,54375029	5,53125	5,75	2,79375005	30,4875011	29,7625008	86,2625	25,4875	26,6	35,6375	29,44125	36,9125	30,1625	43,3625	9,66733456	6,88552666	35,6756744	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:26:36.233
14	5,42500019	5,5687499	5,5625	5,7750001	0,075	30,7000008	29,8625011	86,15	25,5	26,6	35,6625	29,44125	36,6125	30,1875	43,2625	9,67121124	6,88552666	35,5384598	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:27:36.237
15	5,4000001	5,55000019	5,51250029	5,7437501	0,06875	30,4937496	29,6875	86,1375	25,5625	26,6	35,65	29,4275	36,7875	30,225	43,1625	9,67509079	6,881598	35,2671738	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:28:36.233
16	5,32500029	5,4937501	5,4749999	5,7437501	9,08125019	30,5187511	29,4375	86,2125	25,55	26,625	35,625	29,4	36,925	30,225	43,2875	9,68285847	6,88552666	35,5384598	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:29:36.233
17	5,375	5,53125	5,51250029	5,76250029	2,85625005	30,4187508	29,7375011	86,175	25,575	26,6125	35,6375	29,4	36,1125	30,225	43,0375	9,66733456	6,881598	35,5384598	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:30:36.233
18	5,35625029	5,5250001	5,5062499	5,73125029	2,83124995	30,4125004	29,6499996	85,5	25,5875	26,6375	35,6375	29,41375	36,1875	30,225	42,9	9,78499031	6,88552666	35,6756744	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:31:36.233
19	5,33125019	5,51250029	5,48750019	5,73125029	2,8062501	30,4250011	29,6750011	85,55	25,6125	26,65	35,675	29,41375	36,525	30,2625	42,9375	9,82484722	6,88552666	35,402298	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:32:36.233
20	5,34375	5,51875019	5,4937501	5,70625019	2,8125	30,6125011	29,8999996	84,7125	25,65	26,675	35,6625	29,46875	36,6875	30,2625	42,9125	9,67509079	6,88552666	35,2671738	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:33:36.247
21	5,4124999	5,5562501	5,55000019	5,78125	2,82500005	17,2312508	17,84375	84,4125	25,6875	26,6375	35,8125	29,52375	36,6375	30,2875	43,125	9,66733456	6,88552666	35,6756744	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:34:36.230
22	5,34375	5,51250029	5,4937501	5,73125029	2,8125	30,40625	29,6750011	84,4625	25,675	26,675	35,9125	29,55125	37,075	30,325	43,225	9,66346169	6,88552666	35,6756744	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:35:36.233
23	5,35625029	5,5250001	5,5062499	5,7437501	2,8062501	30,6187496	29,7312508	84,5	25,7375	26,6625	36,075	29,55125	37,1625	30,35	43,45	9,66346169	6,88552666	35,6756744	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:36:36.230
24	5,4312501	5,58125019	5,57500029	5,80000019	0,06875	0,075	0,05	84,95	25,7375	26,675	36,05	29,565	37,025	30,3875	43,45	9,66733456	6,88552666	35,6756744	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:37:36.230
25	5,45000029	5,59375	5,59375	5,8187499	0,075	0,125	0,05	85,7875	25,775	26,775	35,9625	29,55125	37,4	30,4125	43,3875	0	0	35,5384598	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:38:36.230
26	5,4375	5,59375	5,5875001	5,83125019	0,075	0,13125001	0,05	86,025	25,775	26,7375	35,9125	29,55125	37,9375	30,4125	43,5375	0	0	35,6756744	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:39:36.230
27	5,48125029	5,61250019	5,6187501	5,8499999	0,075	0,125	0,05	85,6375	25,8125	26,7625	36,4	29,57875	37,875	30,45	43,575	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:40:36.230
28	5,48125029	5,61250019	5,61250019	5,8499999	0,075	0,125	0,04375	85,45	25,7625	26,7375	36,5375	29,60625	37,8875	30,475	43,7375	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 12:41:36.230
29	0,0625	-0,05	0,10625	0,05	0,075	0,11875	9,23750019	41,7625	27,65	26,9	35,325	29,71625	39,025	30,4625	46,0125	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:08:35.227
30	0,0625	-0,05	0,10625	0,05	0,075	0,11875	0,04375	41,0625	27,875	26,9	35,3125	29,74375	38,6625	30,4625	45,6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:09:35.233
31	0,0625	-0,05	0,10625	0,05625	0,08125	0,125	0,05	40,325	28,0375	26,9	35,25	29,7575	38,1625	30,4375	45,55	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:10:35.227
32	0,0625	-0,05625	0,10625	0,05	0,08125	0,11875	0,04375	39,725	28,15	26,9125	35,2375	29,8125	38,05	30,4	45,2625	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:11:35.227
33	0,0625	-0,05625	0,10625	0,05	0,08125	0,125	0,04375	39,2125	28,225	26,925	35,2125	29,84	37,8125	30,375	45,525	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:12:35.230
34	0,0625	-0,05	0,10625	0,05625	0,08125	0,125	0,04375	38,65	28,2875	26,95	35,175	29,84	37,3125	30,375	45,3625	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:13:35.227
35	0,0625	-0,05625	0,10625	0,05	0,08125	0,125	0,04375	38,175	28,325	26,975	35,1875	29,84	37,3125	30,375	45,1375	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:14:35.230
36	0,0625	-0,05	0,10625	0,05	0,08125	0,11875	0,05	37,775	28,3	26,9375	35,175	29,84	36,9125	30,4	45,1125	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:15:35.227
37	0,0625	-0,05625	0,10625	0,05625	0,08125	0,11875	0,04375	37,4	28,425	26,95	35,15	29,82625	37,1625	30,4	45,2375	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:16:35.230
38	0,0625	-0,05625	0,10625	0,05	0,08125	0,11875	0,05	37,0125	28,4875	26,9375	35,1375	29,8675	37,0125	30,4	44,9875	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:17:35.227
39	0,0625	-0,05	0,10625	0,05	0,08125	0,11875	0,05	36,6875	28,5375	26,9375	35,0875	29,88125	36,6125	30,4375	44,6375	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:18:35.230
40	0,0625	-0,05625	0,10625	0,05625	0,08125	0,11875	0,04375	36,4375	28,5125	26,925	35,0875	29,90875	36,1125	30,4375	44,45	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:19:35.227
41	0,0625	-0,05625	0,10625	0,05	0,08125	0,125	0,04375	36,1625	28,55	26,9125	35,1375	29,88125	36,7375	30,4625	44,8625	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:20:35.227
42	0,0625	-0,05	0,10625	0,05	0,08125	0,125	0,05	36,3125	28,525	26,9125	35,1375	29,84	36,875	30,5	44,45	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:21:35.227
43	0,0625	-0,05625	0,10625	0,05	0,08125	0,11875	0,04375	36,3125	28,5125	26,9125	35,125	29,74375	36,9625	30,5	44,325	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:22:35.227
44	0,0625	-0,05	0,10625	0,05	0,075	0,11875	0,04375	36,2875	28,4625	26,9375	35,1	29,71625	37,1875	30,5375	44,0375	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:23:35.227
45	0,0625	-0,05	0,10625	0,05	0,08125	0,11875	0,04375	36,4	28,4125	26,9125	35,0625	29,6475	37,25	30,5375	44,0125	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:24:35.227
46	0,0625	-0,05	0,10625	0,05	0,08125	0,125	0,05	36,35	28,325	26,9	35,0375	29,60625	37,1875	30,5625	44,3875	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:25:35.227
47	0,0625	-0,05	0,10625	0,05	0,08125	0,11875	0,05	36,4	28,2625	26,875	34,9875	29,57875	37,3125	30,6	44,425	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:26:35.227
48	0,0625	-0,05	0,10625	0,05	0,08125	0,11875	0,04375	36,5375	28,225	26,8875	34,9875	29,52375	37,5125	30,6375	44,2625	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:27:35.253
49	0,0625	-0,05	0,10625	0,05	0,08125	0,125	0,05	36,5375	28,1875	26,85	34,925	29,49625	37,75	30,6375	44,425	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2018-03-14 13:28:35.227

Prueba de volcado delos datos capturados por los sensores en la base de datos

GENERALITAT VALENCIANA | TOTS A UNA veu
IVACE INSTITUT VALENCIÀ DE COMPETITIVITAT EMPRESARIAL
UNIÓN EUROPEA Fondo Europeo de Desarrollo Regional
Una manera de hacer Europa
"Proyecto cofinanciado por los Fondos FEDER, dentro del Programa Operativo FEDER de la Comunitat Valenciana 2014 - 2020"