Gobierno del Principado de Asturias
Consejería de Economía y Empleo

Unión Europea
Fondo Europeo
de Desarrollo Regional

Resultados Potenciación y Refuerzo

Línea especialización análisis inteligente de datos

En Gijón , a 31 de Diciembre de 2017

1. MEMORIA TÉCNICA

2016

Hito 1: Diseño de la arquitectura

Para dar respuesta a las necesidades funcionales que se espera cubrir con los activos generados a partir de la línea de actuación SCALARE: Tecnologías de la Información aplicadas al entorno industrial, así como a los requisitos concretos que puedan venir determinados por los diferentes proyectos a desarrollar bajo la misma, se llevará a cabo un diseño arquitectónico modular, genérico y de alto nivel, alineado tanto con las estrategias planteadas por el Emerging Technology Group de la Application Developers Alliance, presentadas en el documento Internet of Things: Manufacturing IoT from the Factory Floor1, como por el Industrial Internet Consortium y su Arquitectura de Referencia para el Internet Industrial, presentado en el documento Industrial Internet Reference Architecture2, y que ha de tener en cuenta el propósito limitado del presente proyecto frente a las características totales definidas en dichas estrategias.

Se define, por tanto, como Internet Industrial a un Internet de cosas, máquinas, computadoras y gente, que habilita la operativa inteligente en los sistemas industriales, haciendo uso de técnicas avanzadas de análisis de datos. Por lo general, este escenario implica la convergencia entre el ecosistema global de la industria y la computación avanzada, mediante sensórica omnipresente y redes con conectividad ubicua.

En los últimos años la barrera que separaba el mundo TI (Tecnologías de la Información) y el mundo TO (Tecnologías de la Operación) se está difuminando. El producto final de este proyecto tiene como objetivo en gran medida ayudar a facilitar la convergencia de ambos modelos. Una de las principales diferencias es la tecnología predominante en cada entorno. Mientras que en el entorno industrial hablamos de sensores, controladores, actuadores, etc.; en el corporativo hablamos de bases de datos, gestor documental, etc. Por ello, el conocimiento tecnológico que poseen los perfiles de cada entorno es totalmente diferente y supone un gran distanciamiento entre ellos.

[1] *Documento disponible aquí: http://www.appdevelopersalliance.org/internet-of-things/manufacturing/*
[2] *Industrial Internet Reference Architecture del Industrial Internet Consortium: http://www.iiconsortium.org/IIRA-1-7-ajs.pdf*

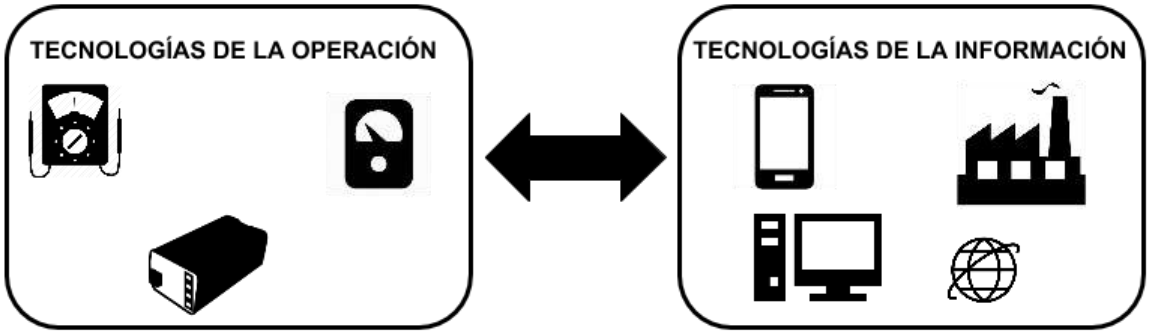

Figura 1: Tecnologías de la Operación y de la Información

La arquitectura conceptual diseñada proporciona un punto de partida estructural, que identificará y definirá pertinentemente los principales componentes del sistema, habilitando un marco de referencia a partir del cual se podrán seleccionar las tecnologías y los módulos software concretos que se habrán de utilizar y/o desarrollar.

T1.1 - Diseño de la arquitectura física.

La arquitectura lambda plantea una solución en la que conviven el procesamiento y presentación de datos en tiempo real con un análisis de los históricos de información recogidos. Teniendo en cuenta las necesidades del proyecto, resulta ser la aproximación más adecuada al diseño del sistema.

En este tipo de arquitecturas se plantean dos enfoques distintos: diferenciado y unificado. El primero considera el procesado y análisis de la información en tiempo real un módulo completamente independiente de aquel que utiliza los datos como un conjunto y realiza análisis offline. Los datos provenientes de ambos módulos son almacenados en sistemas independientes. Por su parte, el enfoque unificado, no contempla la independencia de ambos módulos, estableciendo uno a continuación del otro en la cadena de procesado.

Dadas las necesidades del proyecto, esta última opción, la arquitectura lambda unificada, es la que mejores prestaciones ofrece en su desarrollo. Una vez determinada la arquitectura de alto nivel, se procede a seleccionar las herramientas concretas con las que se cubrirán los requisitos de procesamiento en tiempo real y por lotes.

T1.2 – Diseño de la tecnología para el tratamiento masivo de datos

En cuanto al diseño de la arquitectura servidora Big Data, existen numerosas aproximaciones y enfoques para su estructuración y puesta en escena. Para el desarrollo del proyecto se valoraron tres opciones diferentes:

- Arquitectura kappa
- Arquitectura lambda diferenciada

- Arquitectura lambda unificada

La primera, la arquitectura kappa, plantea un sistema en el cual toda la información es procesada y presentada en tiempo real. No existe como tal un almacenamiento de grandes volúmenes de datos que puedan ser analizados a posteriori. Dado el escenario del presente proyecto este tipo de diseño fue descartado, puesto que se pretende realizar un análisis inteligente de los datos recogidos de las diferentes fuentes que conforman el sistema.

El **módulo de análisis de datos** en tiempo real es el encargado de procesar la información recibida desde la sensórica para realizar los cálculos necesarios con las medidas adquiridas. Este módulo se basa en el sistema de computación distribuida Apache Storm[3].

Apache Storm es un sistema de computación distribuida en tiempo real. Storm está especialmente diseñado para procesar streams de datos continuos de un modo fiable, realizando en el ámbito del procesamiento en tiempo real un trabajo análogo al expuesto por Hadoop[4] en el ámbito del procesamiento por lotes. Se caracteriza por:

- **Rapidez**: Un sistema Storm puede procesar millones de tuplas por segundo, dentro de un nodo particular.
- **Escalabilidad**: Las topologías de Storm son inherentemente paralelas y pueden correr sobre clústeres de máquinas. Las diferentes partes de una topología pueden escalar individualmente, mediante la adaptación de su paralelismo. El comando de rebalanceo permite ajustar dicho paralelismo en tiempo real.
- **Tolerancia a fallos**: Cuando uno de los procesos encargados de recoger y ejecutar los trabajos asignados *(worker)* se cae, automáticamente la plataforma es capaz de relanzarlo. Si un nodo se cae, los workers serán relanzados automáticamente en otro nodo. Los demonios Nimbus y Supervisor están diseñados para no tener estado y ser tener una recuperación rápida ante errores. Por lo tanto, si alguno de dichos demonios sufriera una caída, se relanzarán automáticamente.
- **Garantía en el procesamiento de la información**: Storm garantiza el procesamiento total de cada tupla. Las abstracciones básicas de Storm proporcionan una garantía de procesamiento "una-y-una-sola-vez" similar al proporcionado en las arquitecturas de colas. Los mensajes solamente se reemiten cuando hay fallos.

T1.3 – Estrategia Big Data para almacenamiento escalable

Respecto al almacenamiento de la información gestionada en el sistema, se ha de tener en cuenta la particular problemática que supone el almacenamiento de señales continuas de valores en el tiempo.

[3] *Apache Storm: http://storm.apache.org/*
[4] *Apache Hadoop: http://hadoop.apache.org/*

Para dar respuesta a este problema, se ha decidido hacer uso del sistema Apache Cassandra[5], que es una base de datos NoSQL concebida para gestionar grandes volúmenes de datos manteniendo una gran capacidad de escalabilidad que permite ofrecer un elevado rendimiento tanto en escritura como en lectura.

La arquitectura de Cassandra está basada en el uso de un clúster compuesto por varios nodos y el balanceo de la carga entre ellos.

La información se guarda en tablas con un número de columnas que puede ser variable, con una estructura de datos basada en clave/valor. Cobran especialmente importancia las claves, ya que son el elemento principal en torno al que se balancea la carga de datos en el clúster. También son la unidad de datos mínima que permite la recuperación de la información almacenada en la base de datos.

Los datos se almacenarán en una misma "tabla" que permitirá guardar de forma independiente cada una de las variables de todos los elementos. Esta configuración proporciona una gran flexibilidad, ya que cualquier nueva variable proveniente de cualquier elemento puede ser registrada. Esto se consigue mediante el desacoplamiento de los datos de su significado, es decir qué variables pertenecen a cada elemento e instalación. De este modo, Cassandra sólo almacenará los datos mientras que el significado de los mismos estará almacenado en otra base de datos más apropiada para tal propósito.

La estructura de la tabla usada para el almacenamiento de variables tiene las siguientes características:

- Una fila por cada variable y día. Todos los datos leídos para esa variable en un día son almacenados en la misma fila.
- Cada fila tiene una longitud variable, siendo cada columna el par timestamp (clave)/valor_leído (valor).

Esta estructura permite que los datos estén agrupados para una consulta óptima de los mismos. Las consultas para las que ha sido diseñada esta estructura son del tipo: datos en un periodo de tiempo. Las pruebas iniciales desvelan un gran rendimiento: con 40 millones de registros, la recuperación de todos los valores de una variable para un día se sitúa por debajo del segundo.

2017

Hito 1: Diseño de la arquitectura

T1.2 – Diseño de la tecnología para el tratamiento masivo de datos

[5] *Apache Cassandra: http://cassandra.apache.org/*

A continuación, se mencionarán las distintas opciones relativas al procesamiento de datos, con orientación al procesamiento de grandes volúmenes de datos. Los sistemas de procesamiento se presentan agrupados por el tipo de procesamiento que realizan y una descripción de sus características.

Sistemas de procesamiento Batch

El procesamiento por lotes (batch-processing) se realiza en conjuntos de datos estáticos de gran volumen, en los que el resultado completo se proporciona una vez se han procesado todos los datos.

El procesamiento por lotes es indicado para casos de uso en que se requiere analizar grandes volúmenes de datos estáticos, como por ejemplo el análisis de datos históricos. La desventaja de los sistemas de procesamiento por lotes es el elevado tiempo de computación, que no los hace apropiados para casos de uso en que la latencia o el tiempo de procesamiento resulten relevantes.

Como ejemplo, se destaca:

- Hadoop: Apache Hadoop es un framework de procesamiento por lotes. Fue uno de los primeros frameworks de procesamiento Big Data open-source que posibilitaron el procesamiento de grandes volúmenes de datos. El procesamiento de datos por lotes es realizado por MapReduce. Consiste principalmente en dividir un conjunto de datos extraído de HDFS en varios bloques que se distribuyen entre los distintos nodos de un clúster. Cada nodo aplica una serie de operaciones a cada bloque de datos independiente (Map) y los resultados intermedios de estas operaciones son combinados formando el resultado final (Reduce), siendo este almacenado en HDFS

Sistemas de procesamiento en tiempo real

Los sistemas de procesamiento en tiempo real (Stream processing) procesan datos en el momento en que estos son introducidos en el sistema. Estos sistemas requieren un modelo de procesamiento diferente al del procesamiento por lotes. En lugar de definir operaciones que aplicar a un conjunto de datos en su totalidad, se definen operaciones que aplicar a cada dato que entre en el sistema.

Los conjuntos de datos para un sistema de procesamiento en tiempo real son considerados “infinitos”, ya que quedan determinados por la cantidad de datos que ha sido procesado por el sistema en un momento dado y es eternamente creciente.

Como ejemplo principal, se menciona

- Storm: Apache Storm es un framework de procesamiento de streams cuyo principal objetivo es proporcionar baja latencia, siendo uno de los más indicados para el procesamiento de grandes volúmenes de datos en tiempo real. Storm es el framework recomendado para el procesamiento de streams puro con requisitos de latencias muy bajas. Al ser un framework específico de streaming, sería necesaria la utilización de software adicional si fuera necesario realizar procesamiento por lotes.

Sistemas de procesamiento híbridos

Ciertos frameworks de procesamiento pueden procesar datos en lotes o en streams indistintamente. Estos frameworks estandarizan el procesamiento de ambos paradigmas de procesamiento mediante la definición de componentes y APIs comunes al procesamiento streaming y en lotes.

Mientras que los frameworks específicos para el procesamiento streaming o por lotes están más centrados en una tarea concreta, los frameworks híbridos plantean una solución general para el procesamiento de datos de todo tipo. Alternativamente, estos frameworks ofrecen librerías e integración con herramientas externas para la realización de tareas como análisis de grafos o machine learning.

Entre los frameworks de procesamiento híbridos, destacan:

- Apache Spark: Apache Spark es un framework de procesamiento por lotes con funcionalidad para el procesamiento de streams de datos. Basado en ciertos principios de frameworks de procesamiento anteriores como MapReduce, su principal ventaja es la mejora del rendimiento global al realizar el procesamiento en memoria RAM.

  Spark es el framework más indicado para el procesamiento de cargas variadas, bien por lotes o en streams. El procesamiento por lotes ofrece un gran rendimiento aportado por el procesamiento de datos en memoria, y el procesamiento de streams, aunque no el óptimo para minimizar latencias, permite procesar datos en tiempo real con lotes de menos de un segundo.

- Apache Flink: Apache Flink es un framework de procesamiento de streams con funcionalidad adicional para el procesamiento por lotes. Esta funcionalidad es conseguida considerando un lote como un stream de datos finito, siendo posible tratar el procesamiento por lotes como un sub conjunto del procesamiento de streams.

T1.3 – Estrategia Big Data para almacenamiento escalable

Las principales características de los sistemas de almacenamiento para la gestión de grandes volúmenes de datos son los siguientes:

- Escalabilidad: La escalabilidad es la capacidad de un sistema para responder de manera eficiente ante volúmenes de carga variantes

- Flexibilidad: La capacidad de un sistema de almacenamiento para adaptarse a tipologías de datos dinámicas, sin depender de un esquema de datos gijo

- Disponibilidad: La disponibilidad de un sistema de gestión de datos hace referencia a la disponibilidad constante del sistema, aún en casos anómalos como la presencia de errores.

Todas ellas son características proporcionadas por las bases de datos conocidas como NoSQL. A continuación, se presentan loss diferentes tipos de bases de datos NoSQL y los principales exponentes de cada uno de ellos:

Pares clave valor

Las bases de datos clave-valor se modelan en base a dos componentes asociados: claves y valores. Las claves son identificadores únicos asociados a conjuntos de valores. Los valores representan los datos asociados a una clave. El tipo de datos de los valores puede ser variable, desde una cadena de texto a una imagen o un objeto binarios.

Un ejemplo de base de datos de par clave valor:

- Redis: es una base de datos clave-valor de almacenamiento de estructuras en memoria. Además de actuar como base de datos, puede actuar como caché en memoria o como bróker de mensajes. Ofrece mecanismos de replicación asíncrona de datos mediante paradigma maestro-esclavo y permite realizar operaciones atómicas a los valores asociados a claves.

Bases de datos de documentos

Las bases de datos de documentos, utilizan el modelo clave-valor para almacenar datos, pero con diferencias fundamentales con respecto a las bases de datos con pares clave-valor. Una base de datos de documentos almacena datos como documentos, entidades semiestructuradas en un formato como JSON o XML.

La principal diferencia entra las bases de datos relacionales y las bases de datos de documentos es que estas no requieren un schema predefinido. Otra diferencia importante es la posibilidad de crear documentos dentro de documentos o listas de múltiples valores en un mismo documento.

Como ejemplos de bases de datos de documentos se proporcionan:

- MongoDB: Sistema de gestión de bases de datos NoSQL orientado a documentos. MongoDB almacena las estructuras de datos en documentos de formato BSON (documentos tipo JSON), facilitando la integración de datos. MongoDB se basa en una estructura física máster-esclavo, que permite la escalabilidad y la consistencia de los datos en todo momento. La replicación del máster daría también una disponibilidad de datos prácticamente completa

- Couchbase: Sistema de gestión de bases de datos NoSQL del tipo almacén de documentos. Los datos son almacenados en formato de documentos, con una estructura clave: valor que facilitan el acceso a partir de una clave, sin que los documentos requieran un esquema específico.

Bases de datos de familia de columnas

Aunque manteniendo parecidos con bases de datos relacionales y compartiendo conceptos como las filas y columnas, las bases de datos de familia de columnas (column family) presentan varias diferencias fundamentales. Una columna representa la unidad básica de almacenamiento en una base de datos de familia de columnas, estando formada por un nombre y un valor. Cuando existe un gran número de columnas, estas pueden agruparse en colecciones de columnas relacionadas, conocidas como familias de columnas.

Una fila está formada por un conjunto de columnas o familias de columnas que no tiene por qué ser

compartido entre todas las filas. De este modo, diferentes filas pueden tener diferente número de columnas.

Como ejemplos de base de datos de familia de columnas destacan:

- Cassandra: Apache Cassandra es un sistema de gestión de bases de datos NoSQL. Dentro de los sistemas NoSQL se clasifica dentro de los sistemas de almacenamiento columnar por par clave-valor. Esto es, los datos se almacenan de manera semi-estructurada con entradas clave-valor donde la clave representa la "columna" a considerar y valor su valor. La estructura clave-valor permite una simplicidad operacional reseñable con lecturas de filas de manera sencilla o de todos los valores de una clave

- Uno de los principales puntos fuertes de Cassandra es el alto rendimiento de lectura y escritura de datos, haciéndola una opción muy interesante para recepción de datos de series temporales.

Por otra parte, Cassandra está basado en una arquitectura descentralizada, donde todos los nodos juegan el mismo rol. Este hecho asegura una disponibilidad absoluta del sistema, pese a errores de cualquiera de los nodos. Sin embargo, y al no existir un nodo maestro de control, se pierde la consistencia absoluta del sistema. Por lo tanto, Cassandra se sitúa en el conjunto AP dentro del Teorema CAP.

- HBase: Apache HBase es un sistema NoSQL basado en almacenamiento columnar por par clave-valor. HBase está desarrollado sobre Hadoop y utiliza los conceptos de BigTable desarrollado por Google. HBase opera sobre la capa de almacenamiento proporcionada por HDFS (Hadoop Distributed File System), y depende de otros servicios propios de la distribución de Hadoop como puede ser Zookeeper.

Bases de datos de grafos

En vez de modelar datos utilizando columnas y filas, una base de datos de grafos utiliza estructuras llamadas nodos y relaciones (formalmente vértices y ejes). Un nodo representa un objeto con un identificador y un conjunto de atributos. Una relación es un enlace entre dos nodos que contiene un conjunto de atributos sobre la relación.

Las bases de datos de grafos están diseñadas para modelar proximidad entre objetos. Cada nodo en una base de datos contiene punteros a objetos adyacentes en la base de datos. Esto favorece un alto rendimiento en operaciones que requieren recorrer caminos en un grafo.

Como ejemplos de base de datos de grafos destaca:

- Neo4J: Neo4j es un gestor de bases de datos de grafos desarrollada por Neo Technology. Neo4j se está convirtiendo en la tecnología más utilizada para el almacenamiento de grafos y está sustentada por una gran comunidad que sirve de apoyo para su correcta utilización.

Hito 2: Algoritmos de tratamiento de datos

El sistema de procesamiento de datos desarrollado agrupa los algoritmos en dos categorías principales: Procesamiento en tiempo real y procesamiento en por lotes (batch). Estas categorías referencian los paradigmas de procesamiento principales asociados al procesamiento Big Data, y también determinan las tecnologías a aplicar en cada uno de ellos.

Dentro de cada categoría de procesamiento, diferenciamos una serie de subgrupos, atendiendo a la funcionalidad general que cumple cada uno de los algoritmos desarrollados como parte de la arquitectura de SCALARE. A continuación, se resume el objetivo de cada uno de estos subgrupos, que serán tratados con detalle en posteriores apartados.

La siguiente imagen muestra las agrupaciones definidas, que serán desarrolladas en posteriores apartados.

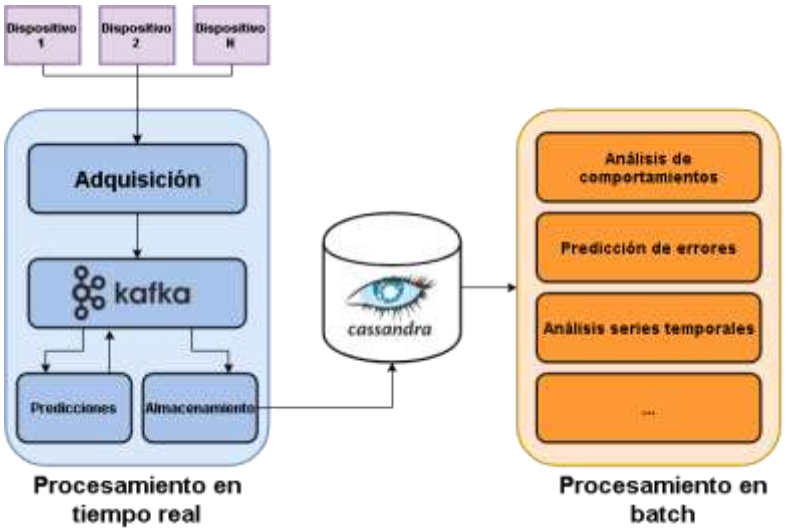

Ilustración 1:Esquema procesamiento datos SCALARE

T2.1 – Algoritmos de tratamiento heterogéneo de datos en tiempo real

Los componentes desarrollados como parte del sistema de procesamiento de datos en tiempo real se agrupan en las siguientes categorías:

Adquisición

Formado por el conjunto de procesos encargados de obtener la información de un conjunto de dispositivos a monitorizar y de introducirla en la arquitectura para facilitar su procesamiento por el resto de rutinas en tiempo real. Siendo indefinido el número de dispositivos para un despliegue específico, estos procesos son implementados utilizando tecnologías BigData que garanticen la escalabilidad global del sistema de adquisición de datos.

Como parte de la arquitectura implementada, todos los servicios de adquisición de datos vuelcan los datos leídos de los dispositivos en Apache Kafka, un sistema de colas distribuido.

Predicciones

Conjunto de procesos encargados de realizar predicciones en tiempo real sobre los datos de dispositivos. Estas predicciones se realizan aplicando modelos machine learning entrenados previamente mediante los módulos de análisis por lotes. Mientras que el entrenamiento de modelos machine learning es demasiado costoso computacionalmente como para realizarse en tiempo real, la aplicación de modelos a nuevos conjuntos de datos sí es posible.

Este subgrupo también englobaría cualquier algoritmo y modelo machine learning que se aplique a datos en tiempo real, como la generación de alarmas.

Almacenamiento

Rutinas encargadas de la persistencia de datos de manera permanente. Mientras que los servicios de adquisición de datos publican los resultados en Kafka, sólo se encuentran en el sistema de manera temporal, mientras son leídos por otros procesos en tiempo real. Los servicios de almacenamiento leen todos los datos publicados en Kafka y los almacenan en Cassandra, una base de datos NoSQL especializada en el almacenamiento de series temporales.

La persistencia de las series temporales en una base de datos facilita el acceso a los datos introducidos en el sistema por parte de servicios adicionales, como los algoritmos de procesamiento batch.

T2.2 – Algoritmos de tratamiento heterogéneo de datos lotes (batch)

Debido a la gran cantidad de datos a considerar y su recepción en tiempo real, será necesario el uso de tecnologías Big Data. Por tanto, se ha decidido utilizar Apache Spark para realizar la predicción de del comportamiento de la temperatura ambiente como serie temporal.

En la tabla siguiente se presenta una comparativa de los modelos óptimos para cada uno de los casos evaluados. Se puede concluir, que en este caso el modelo GBT es el que mejor comportamiento tiene, ya que posee un error cuadrático medio muy inferior al resto tanto en el conjunto de entrenamiento como en el conjunto de test.

Tabla 1. Resultados obtenidos por los diferentes modelos de regresión

Algoritmo	Variables de entrada	Δt	RMSE Train	RMSE Test
Regresión lineal	**Temperatura ambiente** **Irradiancia** **Mes** **Hora** **Minutos** **Segundos** **Presión atmosférica** **Velocidad del viento** **Humedad relativa**	**300**	**1,101**	**1,273**

Árbol de regresión **profundidad: 8**	**Temperatura ambiente** **Irradiancia** **Mes** **Hora** **Minutos** **Segundos** **Presión atmosférica** **Velocidad del viento** **Humedad relativa**	**600**	**0,854**	**1,267**
Random Forest **árboles: 300** **profundidad: 7**	**Temperatura ambiente** **Irradiancia** **Mes** **Hora** **Minutos** **Segundos** **Presión atmosférica** **Velocidad del viento** **Humedad relativa**	**600**	**0,914**	**1,224**
GBT **profundidad: 4** **iteraciones: 100**	**Temperatura ambiente** **Irradiancia** **Mes** **Hora** **Minutos** **Segundos** **Presión atmosférica** **Velocidad del viento** **Humedad relativa**	**300**	**0,695**	**1,082**

2. RESULTADOS PREVISTOS Y RESULTADOS CONSEGUIDOS

Durante el periodo de tiempo que cubre este informe se han conseguido todos los resultados planificados:

- Se ha diseñado una arquitectura de sistemas, basada en el modelo de microservicios interoperables, que cubre todas las necesidades y requisitos esperados.
- Se ha seleccionado un stack de tecnologías Big Data para la implementación de modelos algorítmicos, tanto en batch como en stream. Actualmente está en desarrollo un conjunto de herramientas para facilitar su reutilización en diversos proyectos.
- Se ha seleccionado un conjunto de tecnologías específicas para el almacenamiento de señales masivas procedentes de sensores, dispositivos y controladores industriales.
- Se ha trabajado en la adaptabilidad a la sensórica por parte de la arquitectura.
- Se está realizando el procesamiento de datos y actuación en tiempo real (streaming), así como el procesamiento de datos en offline (batch) para uso de técnicas avanzadas de análisis de datos.