UNIVERSIDADE DA CORUÑA

POLITÉCNICA

XXI Jornadas de Tiempo Real - JTR2019

COMPUTADORES Y CONTROL

Ferrol, 3-4 de setiembre 2019

CONTRIBUCIONES

JORNADAS DE TIEMPO REAL 20199

ÍNDICE DE CONTRIBUCIONES

Sesión: Planificación de sistemas de tiempo real

Sesión: Sistemas distribuidos y particionados

Sesión: Industria 4.0

Mesa redonda: Proyectos de investigación en desarrollo

Apache Cassandra como Gestor de Persistencia de la plataforma P3forI4 diseñada para la Industria 4.0

Ricardo Dintén, Marta Zorrilla, Patricia López, José Mª Drake

Grupo Ingeniería del Software y Tiempo Real. Universidad de Cantabria

Resumen:

En la actualidad, estamos inmersos en la llamada cuarta revolución industrial o también conocida industria 4.0. Esta tiene por objeto revolucionar la industria de la fabricación y producción gracias a la digitalización de los equipos del entorno industrial, la computación en la nube, la integración de los datos y los avances tecnológicos de los sistemas de producción y fabricación. Las plataformas de tercera generación, basadas en arquitecturas centradas en el dato y la computación en la nube, son propuestas para vertebrar este cambio tecnológico. El grupo de investigación ISTR de la Universidad de Cantabria está desarrollando una propuesta arquitectónica denominada P3forI4 basada en los frameworks del ecosistema Apache. La persistencia es uno de los elementos esenciales en esto entorno. Por ello, la elección de la tecnología de persistencia debe realizarse de acuerdo a satisfacer los requisitos de los casos de uso que se desplieguen en el entorno.

En este trabajo, se describe y evalúa el gestor de bases de datos distribuido escalable Apache Cassandra por medio de la ejecución de un conjunto de pruebas (benchmark) con distintas configuraciones y cargas de trabajo para conocer las condiciones límite de operación. Finalmente se ofrecen consideraciones de configuración y diseño para su uso como sistema de persistencia en aplicaciones reactivas y entornos con consideraciones de tiempo real laxos.

Palabras clave: Benchmark, NoSQL, Cassandra, Industria 4.0

1. Introducción

El término Industria 4.0 hace referencia a la cuarta revolución industrial, una nueva etapa en la organización y el control de la cadena de valor a lo largo del ciclo de vida de un producto [5]. La Industria 4.0, por tanto, persigue la transformación del modelo de automatización tradicional con estructura piramidal a un modelo basado en servicios interconectados que combina la tecnología operacional (OT) y las tecnologías de la información (IT) [6]. Su objetivo es alcanzar la operación requerida con una mejora cualitativa en la automatización y optimización de los procesos industriales, satisfaciendo, además la creciente demanda global de productos personalizados. Este modelo se basa en la ubiquidad y conectividad de los datos, las personas, los procesos, los servicios y los sistemas ciberfísicos que intercambian y explotan la información generada en cada nivel de la arquitectura para conseguir una producción descentralizada y adaptable a los cambios en tiempo real.

En este escenario, un entorno industrial 4.0 estará compuesto de un gran número de fuentes (sensores, aplicaciones, sistemas ciberfísicos, etc.) que generarán un ingente volumen de datos que debe ser almacenado, procesado y analizado con diferentes restricciones temporales para aportar valor. El Big Data y la computación en la nube están actualmente considerados como

habilitadores [6] para la construcción del ecosistema industrial del futuro, ya que son capaces de gestionar la variedad, velocidad y volumen de datos por medio de arquitecturas altamente distribuidas y escalables que pueden redimensionarse de manera dinámica para procesar cargas de trabajo en cuasi tiempo real.

La arquitectura P3forI4 propuesta e implementada por el grupo ISTR está diseñada bajo las siguientes premisas:

- Debe procesar los datos procedentes del entorno de modo reactivo y descentralizado.
- La distribución, heterogeneidad y escalabilidad de los recursos computacionales requeridos debe concebirse con un conjunto de servicios de middleware, con el objetivo de seguir una única estrategia para su gestión.
- La monitorización es una tarea esencial en este tipo de sistemas, puesto que su gestión se basa en estrategias dinámicas definidas por el nivel de uso de recursos y el estado de las operaciones

La arquitectura se formula en base a tres principios básicos (ver Figura 1): El dato como servicio (DaaS), la plataforma como servicio (PaaS) y la monitorización como servicio (MaaS). Actualmente se han definido los siguientes servicios (ver ilustración 2): servicio de serialización proporcionado por AVRO, servicio de distribución proporcionado por Zookeeper, servicio de comunicación proporcionado por Kafka, servicio de planificación proporcionado por Spark, servicio de persistencia proporcionado por MemSQL (en memoria), Apache Cassandra (basado en familias de columnas) y Neo4J(basado en grafos) y servicio de seguridad a través de mecanismos proporcionados por Zookeeper para el acceso a información compartida.

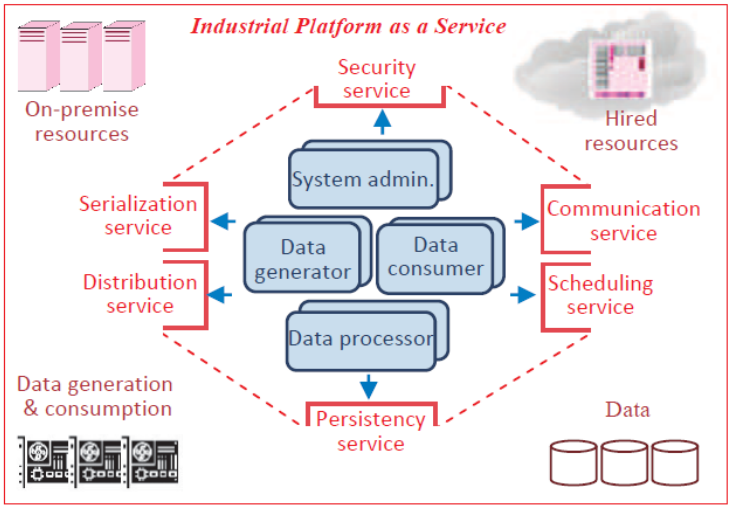

Figura 1. Arquitectura P3forI4

2. Requisitos de persistencia en P3forI4

Los requisitos que la plataforma P3forI4 impone a las tecnologías de persistencia se resumen en:

- Atender el volumen, variedad y velocidad a la que se generan los datos del sector industrial.
- Escalabilidad bajo demanda haciendo uso de recursos contratados en la nube.

- Lectura y escritura de datos con requisitos de consistencia y tiempo real cuasi-estrictos.
- Disponer de estrategias para implantar políticas de seguridad y protección de los datos gestionados.

Los entornos industriales están compuestos por una ingente cantidad de sistemas que generan datos con gran rapidez y, actuadores en los que hay requisitos temporales próximos al tiempo real. Por otro lado, tienen la ventaja de ser entornos poco cambiantes y con un volumen total de dispositivos estable lo que permite realizar un diseño ad-hoc.

A continuación, se evaluará la adecuación de Cassandra para ser integrada en la plataforma.

3. Cassandra: configuración, comportamiento y reglas de diseño

Cassandra [1] es un sistema gestor de bases de datos NoSQL escalable y de código abierto. Está diseñado para manejar grandes cantidades de datos distribuidos en un conjunto de nodos interconectados. Sus principales características son la disponibilidad, la escalabilidad lineal y la simplicidad funcional.

Está basado en estructuras de datos simples, denominadas familia de columnas, para ofrecer flexibilidad y tiempos de respuesta bajos. Estas estructuras son similares a las tablas que se conocen de las bases de datos relacionales, pero a diferencia de estas, cada fila puede tener un número diferente de columnas. Además, no implementa restricciones semánticas ni integridad referencial. Por tanto, hace que las bases de datos sean más flexibles aunque relegando a las aplicaciones el control de la información que almacenan.

Los datos se distribuyen en particiones, que el propio sistema gestiona de manera automática y transparente al usuario, para asegurar que los datos quedan repartidos entre todos los nodos que conforman el sistema. Estos nodos se comunican mediante un protocolo P2P (Peer to peer), es decir, cualquier nodo puede contestar peticiones o realizar la tarea de coordinador. Además se puede configurar el nivel de replicación de los datos para que estén disponibles en varios nodos y mejorar la tolerancia ante fallos del sistema. Esta arquitectura favorece la disponibilidad y los tiempos de respuesta bajos, ya que no es un único nodo el que tiene que recibir todas las peticiones.

3.1 Benchmark

Para poder extraer criterios de configuración y diseño, se realizaron pruebas bajo diferentes configuraciones y cargas de trabajo. Se agruparon en dos apartados:

- Pruebas de rendimiento y escalabilidad dirigidas a analizar el impacto de añadir nodos al clúster, así como su caída y restauración y evaluar el efecto de la replicación y los niveles de consistencia en el tiempo de respuesta del sistema. Se ejecutaron con el sistema YCSB [3].
- Pruebas para determinar estrategias de diseño. Estas pruebas pretenden dilucidar los costes de rendimiento relacionados con el diseño de las tablas que almacenan la información en Cassandra. Estas se realizaron con el software Cassadra-stress [2].

El análisis se realizó en base a dos parámetros: i) *latencia* que es el tiempo de respuesta de una petición, es decir, desde que se ejecuta la sentencia hasta que el sistema la confirma y ii) *throughput*, el número de operaciones por segundo que atiende el gestor.

Los casos de prueba se definieron combinando diferentes valores par los siguientes parámetros:

N° de nodos: 1 y 3

Número de filas: 1.000.000, 33.000.000

Tamaño de filas: 24Bytes, 100bytes

Factor de replicación (Fr): 1 y 3

Nivel de confirmación (Nc): 1, 2 y 3

A partir de combinar estos valores se especificaron las siguientes configuraciones:

conf 0: 1.000.000 filas, 24 bytes, Fr=1, Nc=1

conf 1: 1.000.000 filas, 100 bytes, Fr=1, Nc=1

conf 2: 33.000.000 filas, 24 bytes, Fr=1, Nc=1

conf 3: 1.000.000 filas, 24 bytes, Fr=3, Nc=2

conf 4: 1.000.000 filas, 24 bytes, Fr=3, Nc=3

Para las pruebas de diseño, se diseñó una base de datos, cuyo modelo conceptual se muestra en la Figura 2, que recoge las medidas leídas por dispositivos alojados en diversas máquinas. Se establecieron tres preguntas que permitían observar el efecto en rendimiento en función de las columnas que formaban parte de la partition y clustering key, que son:

Q1: Obtener una máquina y listar todas las máquinas.

Q2: Listar los dispositivos de cada máquina ordenados por localización.

Q3: Recoger todas las medidas tomadas por cada dispositivo por cada parámetro y hora de cada día.

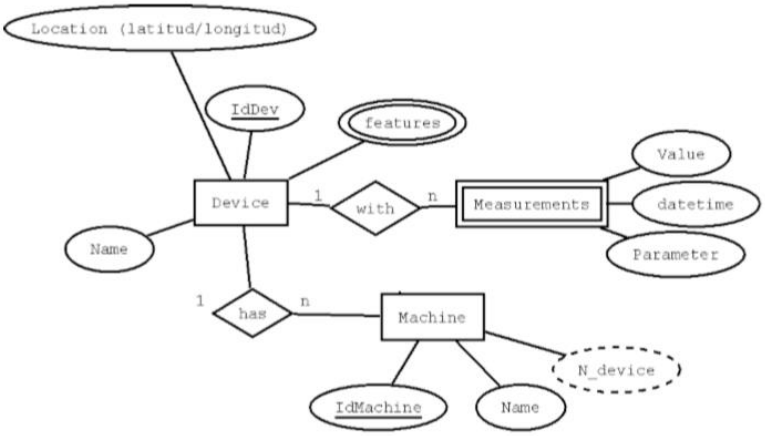

Figura 2. Modelo conceptual de la BD IoT

Debido a la extensión que supondría mostrar todas las gráficas generadas a partir de las pruebas, se indica al lector que están disponibles en [4].

A modo de ejemplo se muestra en la Figura 3 el comportamiento del sistema cuando se escala.

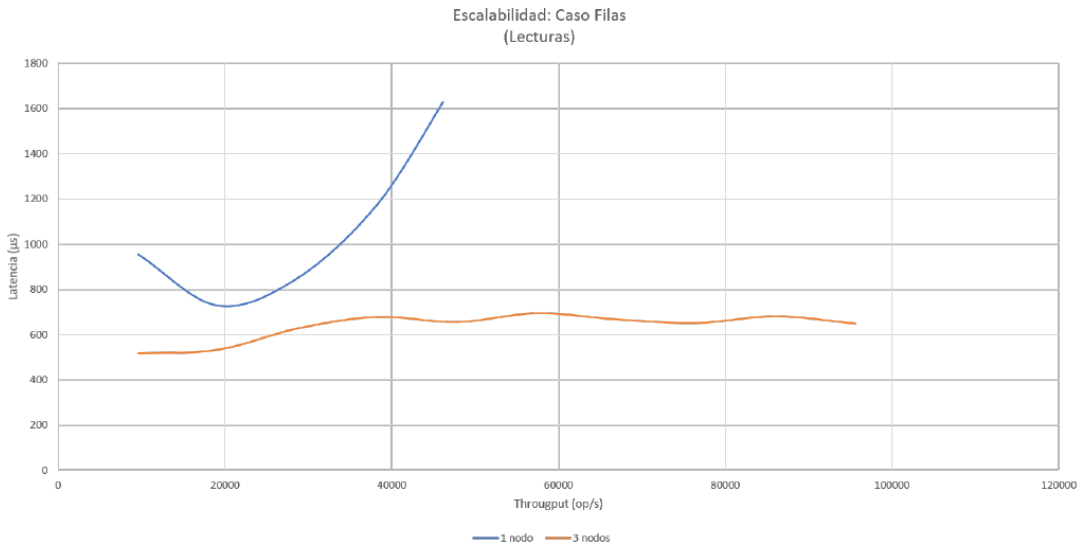

Figura 3. Escalabilidad en lecturas

3.2 Reglas de configuración y diseño

A tenor de la experimentación realizada, a continuación se resumen consideraciones de diseño para hacer uso de Cassandra en un entorno Big Data general y en un entorno I 4.0 con requisitos de tiempo real *soft*:

- **Entornos Big Data**

Cassandra escala linealmente, por ello, el número de nodos a conectar en un clúster se puede estimar a partir del número de operaciones por segundo que soporta un nodo con la latencia máxima permitida y luego, aumentar el número de nodos hasta el número de operaciones a soportar sobre el caso base.

Como método para establecer la topología del clúster es recomendable utilizar el GossipingPropertyFileSnitch, ya que permite añadir nodos con facilidad. En caso de usar otro método, sería necesario modificar el fichero cassadra-topology.properties de cada nodo manualmente.

A la hora de diseñar la base de datos, hay que tener en cuenta las consultas que el sistema debe atender, si son preferentemente de consulta o de actualización y el crecimiento previsto de las tablas con objeto de seleccionar la Primary Key adecuadamente. Se ha de tener en cuenta que:

- Cassandra tiene menores tiempos de respuesta cuando la Primary Key solo está compuesta por un campo, y por tanto, las consultas son por la partition key.
- Conviene diseñar tablas que crezcan más en columnas que en filas. Por lo que la definición de tipos de usuario es una buena alternativa.
- Cassandra tiene un límite de 2 billones americanos de valores por partición, por lo que convendrá incluir campos en la partition key para aquellas tablas que tengan que almacenar volúmenes ingentes de datos.

El nivel de consistencia también se ha de tener presente. Es importante para garantizar la consistencia de la información pero tiene un coste que se multiplica casi linealmente con el número de nodos que deban confirmar. Se ha de llegar a un *trade-off* entre rendimiento e integridad.

El nivel de replicación también es importante e incluso imprescindible para garantizar la disponibilidad y recuperabilidad de los datos. Sin embargo, hay que tener en cuenta que esto exige un mayor tráfico de red y de almacenamiento, ya que se generan datos redundantes y comunicación constante entre los nodos.

Cassandra ofrece la posibilidad de reconfigurar el clúster sin necesidad de parar la actividad, esto permite aumentar el rendimiento de forma sencilla en momentos puntuales de sobrecarga.

- **Entornos I4.0**

Debido a las restricciones temporales que se han de cumplir en los entornos industriales, es necesario reducir al mínimo la sobrecarga impuesta por la red. Para ello se ha de ubicar los datos en el nodo más cercano a los consumidores del mismo con objeto de reducir la latencia.

Cassandra no está diseñado para funcionar de esta manera, pero se han encontrado dos alternativas para solucionar esto:

- Crear un campo *bucket* ajeno a la problemática, que permita controlar el valor de la partition key.
- Crear una nueva implementación de la interfaz IPartitioner para sustituir al partitioner por defecto de Cassandra, el MurMur3Partitioner.

La primera de las opciones se puede llevar a cabo gracias a la herramienta *nodetool* que incorpora Cassandra. Con la opción *getendpoints* se puede conocer la dirección IP del nodo que es responsable de las filas con esa partition key. La segunda opción, consiste en implementar la interfaz IPartitioner en Java para que el particionado se haga haciendo uso de un algoritmo propio. Cuando se ha finalizado se debe indicar en el fichero cassandra.yaml donde se encuentra la implementación del partitioner y, en caso de que ya hubiera datos en el sistema, se ha de vaciar por completo, reconfigurar y volver a cargarlos.

Otro aspecto relevante a considerar por su impacto en la latencia es el nivel de confirmación. Como en el entorno industrial es importante la consistencia de los datos y Cassandra solo proporciona consistencia eventual, lo más correcto es aumentar el nivel de consistencia requerido, de manera que se cumpla que la suma del nivel de consistencia de lecturas y escrituras sea igual o mayor que el factor de replicación + 1. Esto garantiza lo que se conoce como consistencia fuerte. Como hay que añadir cierto nivel de consistencia, se puede priorizar un tipo de operación frente al otro. En sistemas IoT o de I4.0 se generan datos con gran velocidad, por tanto, es más importante que se conteste rápido a las escrituras y no tanto a las lecturas.

Por último, se debe tener en cuenta el volumen total de los datos que se van a manejar y durante cuánto tiempo se van a persistir. En entornos industriales, que no aumentan la cantidad de dispositivos, ni crecen con frecuencia, es relativamente sencillo. Esto es importante, ya que al ser el volumen de datos elevado existe el riesgo de saturar una partición, llegando al límite de 2 billones americanos de valores. Por tanto, se debería hacer una estimación que permita evitar este problema, sin hacer que el número de particiones sea tan grande que no se pueda controlar y ubicar como se desea.

4. Conclusiones

Este trabajo resume las características del gestor de persistencia NoSQL Apache Cassandra y ofrece pautas para su configuración y utilización en un entorno industrial para aplicaciones reactivas, descentralizadas y con requisitos temporales laxos extraídas a partir de una serie de pruebas de rendimiento bajo diferentes condiciones de carga y diseño.

Agradecimientos

Este trabajo ha sido financiado por el Gobierno de España a través del proyecto TIN2017-86520-C3-3_R.

Referencias

[1] About apache cassandra | apache cassandra 3.0, último acceso el 06/06/2019. https://docs.datastax.com/en/cassandra/3.0/cassandra/cassandraAbout.html

[2] The cassandra-stress tool | apache cassandra 3.0, último acceso el 04/06/2019. https://docs.datastax.com/en/cassandra/3.0/cassandra/tools/toolsCStress.html

[3] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, y R. Sears. Benchmarking cloud serving systems with ycsb. In Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC '10, pages 143–154, New York, NY, USA, 2010. ACM. http://doi.acm.org/10.1145/1807128.1807152

[4] Ricardo Dintén. Especificación y ejecución de benchmarks sobre el gestor de datos distribuido y escalable Cassandra para aplicaciones de uso intenso de datos en la Industria 4.0. Trabajo fin de grado en Ingeniería Informática. Julio 2019

[5] K.-D. Thoben, S. Wiesner, y T. Wuest. industrie 4.0 and smart manufacturing a review of research issues and application examples. International Journal of Automation Technology, 11(1):4–16, 2017.

[6] N. Velásquez, E. Estévez y P. Pesado. Cloud computing, big data and the industry 4.0 reference architectures. Journal of Computer Science and Technology, 18(03):e29, Dec. 2018. http://journal.info.unlp.edu.ar/JCST/article/view/1151