SOLID: una Arquitectura para la Gestión de Big Semantic Data en Tiempo Real*

Mario Arias[1], Carlos E. Cuesta[2], Javier D. Fernández[3,4], Miguel A. Martínez-Prieto[3]

[1] Digital Enterprise Research Institute (DERI), National University of Ireland, Galway (Ireland)
[2] VorTIC3 Research Group, Dept. de Lenguajes y Sistemas Informáticos II, Universidad Rey Juan Carlos, Madrid (España)
[3] DataWeb Research, Dept. de Informática, Universidad de Valladolid, Valladolid (España)
[4] Dept. de Ciencias de la Computación, Universidad de Chile, Santiago (Chile)
mario.arias@deri.org, carlos.cuesta@urjc.es, {jfergar,migumar2}@infor.uva.es

Resumen La gestión de grandes colecciones de datos (*Big Data*) es un proceso crítico en entornos de explotación en tiempo real ya que las arquitecturas *batch*, que garantizan un comportamiento escalable, ofrecen unos tiempos de respuesta insuficientes para los requisitos de rendimiento que se presentan en dichos entornos. En este artículo se estudia esta problemática, de acuerdo a las necesidades planteadas por aquellos sistemas de información en los que se forman y exponen grandes colecciones de RDF (*Big Semantic Data*) en tiempo real. Nuestra propuesta es una nueva arquitectura (SOLID) que aísla la complejidad de almacenar grandes colecciones de datos y las necesidades específicas de insertar y consultar *Big Semantic Data* en tiempo real. La base tecnológica de SOLID comprende el uso de RDF/HDT para el almacenamiento *auto-indexado* de los datos y tecnología NoSQL para su gestión en tiempo real. Nuestros resultados experimentales muestran la eficiencia de cada una de las capas de datos y su integración mediante dos capas software adicionales que garantizan la escalabilidad de SOLID.

1. Introducción

Aunque *Big Data* es una de las palabras de moda en el panorama tecnológico actual, no existe un consenso en cuanto a su definición. Una de las más aceptadas indica que hablamos de *Big Data* "cuando el tamaño de los datos, por sí mismo, forma parte del problema" [11]. Sin embargo, esta definición sólo considera la dimensión más obvia, el **volumen**, pero excluye la **velocidad** y la **variedad**. La convergencia de estas *tres Vs* [9] define la caracterización primitiva de *Big Data*.

El *volumen* es la dimensión más obvia al caracterizar grandes colecciones de datos creadas para diferentes usos y propósitos. El *almacenamiento* de *Big Data* supone el reto más inmediato, ya que la primera responsabilidad es la de preservar todos los datos generados en el ámbito de actuación del sistema. La decisión de cómo se almacenan los datos tiene, a su vez, un impacto considerable en el rendimiento de los procesos de *recuperación*, *procesamiento* y *análisis* de *Big Data*.

La *velocidad* caracteriza los flujos de datos desarrollados en entornos cada vez más distribuidos. Se pueden distinguir dos tipos: i) flujos de nuevos datos (generados de diferentes maneras y por diferentes fuentes) que deben ser integrados de forma progresiva

* Este trabajo ha sido financiado por el Minist. de Economía y Competitividad: TIN2012-31104 y TIN2009-14009-C02-0.

en los *Big Data* existentes y ii) flujos que contienen los resultados de las consultas y cuyo volumen puede ser potencialmente grande. Por lo tanto, la velocidad describe lo rápido que se generan, demandan y entregan los datos en su entorno de explotación.

La *variedad* se refiere a los diferentes grados de estructura (o falta de ella) que pueden encontrarse en una colección considerada *Big Data*. La colección puede integrar datos procedentes de múltiples fuentes (redes de sensores, *logs* generados en servidores web, redes sociales, datos de origen político, económico o científico, entre otros) y, obviamente, cada una de ellas posee una semántica particular que da lugar a esquemas diferentes que son difícilmente integrables en un modelo único. Por lo tanto, el manejo efectivo de la variedad pasa por encontrar un modelo lógico que facilite la integración de los datos con independencia de su estructura.

La caracterización de las tres Vs plantea una descripción genérica de *Big Data*, pero la definición del volumen merece ser revisada para dar una mayor cobertura a la situación tecnológica actual. En general, existe una tendencia a entender el *Big Data* en escalas de *terabytes*, *petabytes* o *exabytes*. Sin embargo, existen numerosos casos de uso reales en los que unos pocos *gigabytes* de datos pueden ser suficientes para hacer colapsar una aplicación que esté ejecutándose en un dispositivo móvil e, incluso, en un ordenador personal. De acuerdo con esta consideración, en el resto del artículo utilizaremos el concepto *Big Data* para referirnos a cualquier conjunto de datos cuyo tamaño pone al límite la cantidad de recursos disponibles para su almacenamiento y procesamiento en un sistema computacional dado.

Cualquier arquitectura diseñada para la gestión de *Big Data* [3] debe afrontar las tres dimensiones anteriores. Sin embargo, la decisión de cuál de ellas afrontar en primer lugar depende del entorno de explotación final de la arquitectura. Por ejemplo, optimizar el almacenamiento de los datos es un aspecto más crítico para una arquitectura diseñada para un dispositivo móvil que para otra que vaya a ser ejecutada en un servidor de alto rendimiento; la velocidad a la que se recuperan los datos es una prioridad para una arquitectura de tiempo real pero no lo es tanto para una de procesamiento en *batch*. Por lo tanto, una arquitectura para *Big Data* debe priorizar las tres dimensiones anteriores con el objetivo de cubrir, de forma efectiva, los requisitos con los que se diseña.

Nuestro trabajo parte de una precondición clara: el conocimiento que se puede generar a partir de un *Big Data* crece al integrarse más datos en la colección, lo que trae consigo un aumento de su **valor**. *Esta decisión implica la promoción de la variedad frente al volumen y la velocidad* y se materializa en el modelo utilizado para la organización lógica de los datos. Nuestra elección es usar un *modelo de grafo* a través del cual se puede representar una mayor diversidad de datos, facilitando con ello su interrelación y consulta integrada [19]. En la actualidad, la línea de trabajo más avanzada se centra en el uso de RDF (*Resource Description Framework*) [12].

RDF es una de las piedras angulares del proyecto original de Web Semántica [4] cuyos principios básicos se materializan actualmente en la Web de Datos (*Linked Data*). RDF alcanza su máximo potencial cuando se utiliza conjuntamente con vocabularios que describen la *semántica* de los datos, por lo cual se referirá como *Big Semantic Data* [6] cualquier colección de RDF considerada *Big Data*. Es importante resaltar que la elección de RDF como modelo de datos no supone un discriminante en el tipo de sistema software que puede ser desarrollado, pero sí restringe la estructura y el alma-

cenamiento de los datos, así como la forma en la que son accedidos. Por lo tanto, la elección de RDF sí determina *cómo se afrontan las dimensiones volumen y velocidad*.

Nuestra propuesta, SOLID *(Service-OnLine-Index-Data)* es una arquitectura de alto rendimiento para la construcción de sistemas de tiempo real basados en el consumo de RDF. SOLID describe una configuración estructural basada en el aislamiento de las complejidades asociadas con el consumo de *Big Semantic Data* y la gestión de datos en tiempo real. Por un lado, SOLID plantea una representación compacta del *Big Data*. Esto supone almacenar e indexar los datos en espacio comprimido, lo que trae consigo un ahorro importante tanto en recursos de almacenamiento como de procesamiento, mejorando también el rendimiento de la consulta. Las capas *Data* e *Index* implementan estas responsabilidades aprovechando la estructura compacta de RDF/HDT [13,8]. Sin embargo, mantener los datos en forma compacta dificulta su actualización. Por tanto, el procesamiento de las nuevas piezas de RDF, generadas en tiempo real, se lleva a cabo en una tercera capa que ofrece unas altas prestaciones en la actualización de los datos, además de proveer un buen rendimiento en consulta. Esta capa *Online* se materializa sobre tecnología NoSQL. Aunque ambas visiones cumplen de forma efectiva los objetivos particulares con los que se diseñan, SOLID precisa de dos capas adicionales para alcanzar su objetivo global: i) la capa *Merge* aprovecha las propiedades de RDF/HDT para integrar los "*datos online*" en el *Big Semantic Data* mientras que ii) la capa *Service* coordina las operaciones de consulta SPARQL [18] sobre las capas *Index* y *Online*.

El resto del artículo se estructura como sigue. La Sección 2 da una visión general del contexto desarrollado en torno al uso y explotación de RDF. La Sección 3 describe SOLID y la Sección 4 plantea su implementación utilizando diferentes tecnologías desarrolladas en el estado del arte. La Sección 5 presenta algunos resultados iniciales de SOLID y, por último, la Sección 6 concluye sobre los mismos y plantea las líneas de trabajo futuro a desarrollar hasta obtener la implementación final de la arquitectura.

2. Motivación

Esta sección revisa las propiedades básicas de RDF, su impacto actual y algunas de las tecnologías desarrolladas en torno a sus necesidades. Una visión más completa de estos tópicos se puede encontrar en [10]. La sección finaliza con un ejemplo que ilustra los requisitos sobre los que se diseña la propuesta presentada en este trabajo.

2.1. RDF en el Escenario Tecnológico Actual

RDF (*Resource Description Framework*) [12] plantea un modelo de datos de gran simplicidad en el que una *entidad* (también denominada *recurso*) se describe como una terna *(sujeto, predicado, objeto)*. Por ejemplo, la descripción de una transacción (basada en el ejemplo descrito en la Sección 2.3) `check-in#1`, realizada el `02/04/2013` en la estación `station#123`, se puede describir como: `(checkin#1, date, "02/04/2013")` y `(check-in#1, in_station, station#123)`.

Las ternas de una colección RDF forman una estructura de grafo de conocimiento en el que las entidades y los valores se unen mediante arcos etiquetados. Estas etiquetas desarrollan el rol de predicado y definen la semántica de la relación. Por lo tanto, en pos de describir este significado, de una manera lo menos ambigua posible, se recomienda el uso de vocabularios estándar o, de no existir, formalizar vocabularios propios.

El uso de RDF se ha extendido fuertemente gracias a su adopción en proyectos procedentes de campos como la bioinformática, la geografía, o las redes sociales, entre otros. Las colecciones obtenidas en estos proyectos se han expuesto en la Web dando lugar a la nube de datos enlazados que conforma el núcleo del proyecto *Linked Open Data*[5] (LOD). LOD juega un papel crucial en la evolución de RDF [10]. Este proyecto aprovecha la infraestructura de la Web para dotar de identidad global (mediante URIs HTTP) a cada uno de los datos descritos en estas colecciones RDF. Esta decisión garantiza que todo dato es identificable en la Web, de tal forma que dos datos cualesquiera pueden ser enlazados a través de una nueva terna RDF. Por ejemplo, si la estación `station#123` describe la parada de *Canal Street*, la terna `(station#123, location, <http://dbpedia.org/page/Canal_Street>)` permite enlazar nuestra entidad `station#123` con toda la información publicada en DBpedia[6] sobre *Canal Street*, de tal forma que se podrá "navegar" desde la estación a toda la información del lugar.

LOD plantea una perspectiva complementaria de la Web que la presenta como una gran red de hipervínculos entre datos frente a la visión tradicional que describe la Web como una red de documentos (páginas) interrelacionados. Las últimas estadísticas[7] publicadas sobre el tamaño de la nube de LOD (septiembre 2011), indicaban que contenía 31 billones de ternas RDF y más de 500 millones de hipervínculos entre las colecciones expuestas por 295 fuentes diferentes. El tamaño de las colecciones mantenidas por cada uno de los publicadores puede ser particularmente pequeño (estas colecciones se clasifican dentro del *Big Data's long tail*), pero su integración, en una visión parcial o total de esta Web de Datos, sí representa un ejemplo claro de *Big Semantic Data*.

Algunos procesos Web tradicionales (*crawlers*, motores de búsqueda o sistemas de recomendación), junto con dispositivos como las tarjetas RFID, los teléfonos inteligentes y los sensores son fuentes potenciales de RDF como ilustramos en nuestro ejemplo. En su conjunto, dan lugar a la conocida como "*Internet de la Cosas*", donde el paradigma de datos enlazados puede jugar un papel crucial a la hora de integrar estos dispositivos dentro de la infraestructura de la Web [17]. Cada uno de estos objetos del mundo real puede ser identificado mediante una URI y la información que genera se puede estructurar mediante ternas RDF, en las que la URI correspondiente actúa como sujeto. De esta manera, se registra la actividad generada estos objetos y, lo que es más importante, se habilita su interrelación, lo que resulta clave en el desarrollo de proyectos de gran envergadura como, por ejemplo, las ciudades inteligentes (*smart-cities*) [5].

2.2. Tecnologías RDF Básicas

El rol central que juega RDF en la Web de Datos implica el desarrollo de nuevas tecnologías que lo complementan de diferentes maneras. Sin embargo, esta sección no pretende dar una visión completa de este contexto, sólo introducir las tecnologías básicas sobre las que se desarrolla RDF: los formatos propuestos para su serialización y algunas de las propuestas utilizadas para su almacenamiento (*RDF Stores*).

La elección de un determinado formato de representación es un factor importante a la hora de gestionar *Big Semantic Data*, ya que la recomendación RDF [12] describe el

[5] `http://linkeddata.org/`

[6] DBpedia: `http://dbpedia.org`, es una conversión parcial de Wikipedia a RDF

[7] `http://www4.wiwiss.fu-berlin.de/lodcloud/state/`

modelo lógico de datos pero no restringe la forma en la que este es finalmente serializado. RDF/XML [2] es el formato originalmente recomendado para la representación de colecciones semánticas. RDF/XML codifica el grafo RDF como un árbol XML, obteniendo una representación orientada al documento que no resulta aceptable a la hora de serializar *Big Semantic Data*. Además, este tipo de formatos da lugar a representaciones de gran tamaño que limitan las posibilidades de explotación de las colecciones RDF [13]. Algunas otras sintaxis han sido propuestas posteriormente (una revisión de las mismas se puede encontrar en [10, §2.4.2]), pero RDF/HDT[8] [8] fue el primer formato cuyo diseño abordó directamente el problema del tamaño de las representaciones.

RDF/HDT es un formato binario que reduce la verbosidad de la serialización del grafo RDF en pos de obtener una representación procesable de forma automática. Esta capacidad de RDF/HDT facilita la gestión eficiente de *Big Semantic Data* dentro de los flujos de trabajo más comunes de la Web de Datos, siendo un formato ideal para el almacenamiento y el intercambio de RDF (RDF/HDT usa hasta 15 veces menos espacio que las sintaxis tradicionales [8]). La especificación de RDF/HDT plantea, adicionalmente, configuraciones específicas de estructuras de datos compactas que permiten acceder de forma eficiente a la colección en espacio comprimido. Estas configuraciones han sido recientemente extendidas [13] para soportar consulta SPARQL, obteniendo una representación en la que *"los datos son el índice"*. Aunque la técnica resultante (HDT-FoQ) aumenta de forma controlada el tamaño de la representación, se mantiene como la propuesta más compacta dentro del estado del arte y obtiene un rendimiento competitivo respecto al obtenido por técnicas consolidadas como Virtuoso[9] o RDF3X [16], superándolas en diferentes escenarios de consulta.

Virtuoso y RDF3X son dos ejemplos bien conocidos de *RDF Stores*. Virtuoso es una tecnología ampliamente difundida y utilizada en la actualidad, con productos comerciales y *open-source* destinados a la gestión de RDF. Por su parte, RDF3X plantea un trabajo de investigación excelente cuyos resultados experimentales lo sitúan como la técnica de referencia en el área. Ambas técnicas tienen en común el modelo lógico de datos que implementan: RDF y el lenguaje que utilizan para su consulta: SPARQL, aunque cada una de ellas utiliza una aproximación diferente tanto para la organización y almacenamiento de los datos como para el procesamiento y la optimización de las consultas. No obstante, ambas técnicas (al igual que otros RDF Stores como Jena[10]) se pueden clasificar dentro de lo que se viene denominando como bases de datos NoSQL.

Los sistemas NoSQL se diseñan, en general, como soluciones simples con una funcionalidad limitada respecto a la ofrecida por una base de datos relacional. Esta simplicidad permite optimizar el rendimiento de las operaciones primitivas de inserción y/o acceso a datos, garantizando un buen rendimiento global y la escalabilidad de los sistemas construidos sobre ellas. Bajo esta descripción, el uso de RDF Stores parece una aplicación directa al problema tratado en este trabajo. Sin embargo, en la práctica, las necesidades de escalabilidad y de gestión del *Big Semantic Data* en tiempo real chocan entre sí, reduciendo el grado de satisfacción obtenido al usar estas soluciones en el desarrollo de sistemas de información como el descrito en el ejemplo siguiente.

[8] HDT fue reconocido como W3C Member Submission en 2011 [7].
[9] `http://virtuoso.openlinksw.com/` (Virtuoso) – `http://jena.apache.org` (Jena)

2.3. Un Caso de Ejemplo

Cada vez que utilizamos tarjeta del transporte se registra la transacción. Esta puede contener, al menos, un identificador anonimizado, la fecha y la hora, así como la estación en la que se realiza. Analicemos lo que esto supone en la red de metro de Nueva York (NYC), con un total de 468 estaciones, 5 millones de usuarios en días laborables y un total de 1640 millones anuales Esta colección, es un ejemplo claro de *Big Data* y potencial objeto de estudio dado su interés, es decir, susceptible de que sus datos sean consultados, analizados e integrados con otros datos (ej. servicios urbanos próximos a las estaciones, eventos culturales o deportivos que puedan influir en la movilidad, la meteorología en un periodo de tiempo, etc.). Esto da lugar a la necesidad de recoger e integrar, en dicha colección, miles de nuevos registros por minuto.

Al cabo de 5 años, el sistema de información que gestione la colección deberá proporcionar la infraestructura necesaria para almacenar más de 8000 millones de registros en constante crecimiento, y que se consultarán para propósitos varios, tales como la predicción de procesos organizativos y logísticos en el servicio de metro, análisis estadísticos y sociológicos, o la implantación de procesos de *Business Intelligence*. La complejidad inherente a este sistema se puede plantear desde dos perspectivas complementarias. Por un lado, la necesidad de almacenar nuevos datos en tiempo real no puede materializarse en inserciones continuas que, a su vez, implicarían una regeneración constante de los índices utilizados para consulta, degradando notablemente el rendimiento de las tareas ejecutadas en el sistema. Por otro lado, no se puede optar por una inserción en *batch* diferida (por ejemplo, al final del día) ya que para algunos procesos puede ser imprescindible acceder a los últimos datos recogidos en las estaciones.

3. Arquitectura SOLID

La gestión y procesamiento de *Big Data* en tiempo real es un problema abierto y vigente. Según Marz y Warren [14], no hay una única herramienta que ofrezca una solución completa, sino que la solución pasa por definir una propuesta consistente que permita seleccionar una configuración de tecnologías particular con la que satisfacer los requisitos del sistema a desarrollar. En nuestro contexto, se ha de considerar que el *Big Data* tiene una naturaleza *semántica*, lo que hace a su procesamiento dependiente del modelo de datos elegido. Incluso cuando se elige un modelo tan genérico como RDF, éste tiene una influencia decisiva sobre la configuración óptima de la arquitectura.

Nuestra propuesta, SOLID, consiste en una arquitectura diseñada específicamente para el tratamiento de *Big Data* semántico (*Big Semantic Data*) en tiempo real, dónde se asume desde el principio que la información tratada va a ser RDF – sin que esto imponga una representación específica de RDF. La estructura global se ilustra en la Figura 1. En esencia, se trata de una arquitectura *3-tier*, con tres niveles: *Sevice*, *Data* y *Merge*, en la que el nivel central se subdivide a su vez en tres capas (*Online*, *Index* y *Data*), lo que resulta en una arquitectura de cinco capas no estrictamente ortogonales. Obsérvese que aquí se describe únicamente la arquitectura abstracta, mientras que en la sección 4 se proporciona una propuesta tecnológica de implementación. No obstante, se podrían desarrollar otras implementaciones de SOLID utilizando una tecnología diferente, sin perder (la mayoría de) las propiedades que proporciona esta arquitectura.

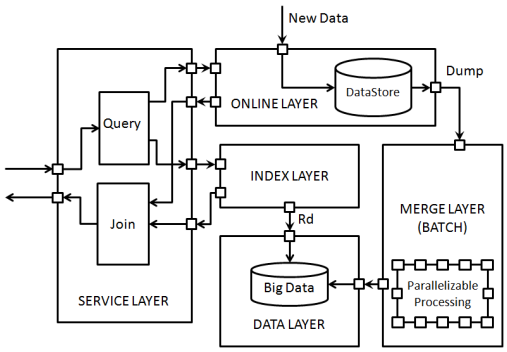

Figura 1. Arquitectura Abstracta SOLID (sin detalles tecnológicos)

La idea central de la arquitectura consiste en separar el almacenamiento principal del sistema (que ha de enfrentar las particularidades del *Big Data*) del procesamiento de los datos recogidos en tiempo real. Es decir, se mantienen dos almacenes de datos: uno es un gran almacén de datos, con todas las propiedades inherentes al *Big Data* (en particular su *inmutabilidad*) y en el otro se reciben y procesan datos en tiempo real. El impacto de esta decisión se aprecia al analizar la forma en la que se accede a la información. El cliente envía sus consultas a través de la capa *Service* que la divide en dos flujos: uno se dirige al almacén de datos principal, al que se accede a través de la capa *Index*, mientras que el otro se dirige a la capa *Online* para acceder a los datos procesados en tiempo real. De este modo, la resolución de una consulta SPARQL comprende una búsqueda en cada uno de los almacenes, de los cuales se recuperan sendos flujos de resultados que finalmente se integran en la capa *Service*, antes de ser entregados al cliente. Por lo tanto, esta división de responsabilidades permite gestionar eficientemente los requisitos de *velocidad*, sin comprometer con ello el rendimiento global de la solución, ya que ninguna de las operaciones implementadas en tiempo real (en la capa *Online*) tiene impacto en el almacén principal.

Obviamente, el almacén de la capa *Online* crece progresivamente, lo que puede dar lugar a una degradación en el rendimiento del procesamiento en tiempo real. Por este motivo, se proporciona una capa adicional de procesamiento por lotes: *Merge*, en la que se vuelca el contenido del almacén de la capa *Online* para su integración en el almacén principal. Esta es una tarea costosa que requiere un uso exhaustivo de recursos de cómputo, por lo cual se implementa como un proceso programado en *batch* que comprometa lo menos posible, el rendimiento global del sistema. Una vez completado el proceso de integración se disparan dos procesos finales: a) las estructuras de indexación consideradas en la capa *Index* se actualizan de acuerdo a los nuevos contenidos del *Big Semantic Data* y b) la capa *Online* descarta los datos ya integrados y que, desde este momento, son accesibles a través del almacén principal.

Capas. Las diferentes capas de SOLID se caracterizan como sigue.

La capa **Data** se responsabiliza de la complejidad subyacente al almacenamiento de *Big Semantic Data*, implementando su propiedad principal: la *inmutabilidad* de los datos. Esto supone que ninguno de los datos que se inserte en esta capa podrá ser modificado a posteriori, de tal manera que su requisito principal es el de ofrecer una forma

efectiva de almacenar los datos modelados en RDF. Como se indicaba en la sección anterior, esta necesidad pasa por elegir un formato de representación de RDF que satisfaga los requisitos de la arquitectura. En nuestro caso, el requisito principal es el de obtener un almacenamiento lo más compacto posible.

La capa **Index** implementa los mecanismos de indexación necesarios para acceder de forma eficiente al *Big Semantic Data* utilizando consultas SPARQL. Estos mecanismos se construyen directamente sobre la capa *Data*, integrando la representación del *Big Semantic Data* como parte del propio sistema de indexación. Este tipo de soluciones se denominan *auto-índices* [15] y permiten representar tanto los datos como las estructuras de indexación en espacio comprimido.

La capa **Online** es la capa superior en la pila de las tres capas de datos. Esta capa puede ser entendida como un "*buffer temporal*" en el que se gestionan los datos generados en tiempo real hasta que se decide su integración (planificada) en el *Big Semantic Data*. Por lo tanto, el requisito principal de esta capa pasa por ofrecer una alta velocidad de inserción que permita acometer los flujos de datos que se puedan presentar. Además, esta capa debe proveer resolución de consulta basada en SPARQL.

La capa **Merge** actúa como nexo entre las capas *Online* y *Data* e implementa la integración de los datos capturados en tiempo real dentro del *Big Semantic Data*. Este proceso tiene el objetivo de controlar el volumen de los datos gestionados por la capa *Online*, dado que un crecimiento excesivo podría provocar una degradación no deseada en la velocidad de los procesos de tiempo real. Por este motivo, de forma periódica se realiza un volcado (*dump*) del almacén de la capa *Online* que se recoge en la capa *Merge*. Dependiendo de la tecnología utilizada en la capa *Online*, el *dump* podría estar serializado en un formato RDF adecuado para la integración o, por el contrario, la capa *Merge* debería transformarlo de acuerdo a sus necesidades. La capa *Merge* toma como entradas los datos procedentes de las capas *Online* y *Data* y a partir de ellos obtiene una versión actualizada del *Big Semantic Data*. Este proceso se realizará en *batch* atendiendo a la gran cantidad de recursos computacionales que se requieren para manejar y procesar este gran volumen de datos. No obstante, el proceso de integración de los datos se puede paralelizar (de forma local e incluso distribuida) en tiempo de implementación, abriendo un gran abanico de posibilidades para su optimización. En general, se considera más eficiente generar un nuevo almacén, en lugar de integrar los nuevos datos en la estructura ya existente, aunque el formato de almacenamiento elegido puede tener una gran influencia en este sentido. En cualquier caso, esta decisión garantiza el servicio normal de nuestra arquitectura, incluso cuando la capa *Merge* está en ejecución. Finalmente, indicar que la capa *Merge* es *separable* del resto de la arquitectura. Obsérvese que es ella misma quién solicita el *dump* a la capa *Online*, y una vez termina su función proporciona la nueva versión del *Big Semantic Data* a la capa *Data*. Por tanto, sólo interactúa con la arquitectura en el momento en el que ella misma lo necesita, de tal forma que podría no estar accesible el resto del tiempo.

La capa **Service** proporciona una API de consulta SPARQL que permite interactuar con los datos gestionados por el sistema de información desarrollado sobre SOLID. El cliente realiza consultas (directamente o a través de una aplicación), que se recogen en un módulo de consulta (*Query*), donde ésta se multiplexa como dos subconsultas, dirigidas respectivamente a las capas *Online* e *Index*. Eventualmente ambas proporcionan

sus flujos de respuesta, de acuerdo a la semántica de la consulta, que son *reunidos* por un módulo de fusión (*Join*) y combinados para obtener la respuesta final. La función de este módulo *Join* puede ser una simple yuxtaposición de ambos flujos, o realizar un procedimiento más elaborado. De hecho, el tratamiento de estos flujos se construye como una instancia del estilo filtro-tubería, por lo que se puede combinar una secuencia de filtros como un *pipeline* de datos que realice un post-procesamiento de los datos recibidos, que puede ir del simple filtrado o la presentación en un formato adecuado, hasta un tratamiento parcial de estos datos con algún propósito específico. Por lo tanto, la capa *Service* actúa como *mediador* [20], permitiendo combinar la capacidad de tiempo real de la capa *Online* con la velocidad de la capa *Index*, sin comprometer cada una de ellas.

En conclusión, la configuración en capas de SOLID garantiza que la visión que se percibe desde el exterior es la de un almacén de datos que provee una interfaz de consulta SPARQL sobre una un *Big Semantic Data* actualizable en tiempo real. En la práctica, el rendimiento de SOLID es equivalente al que se obtendría al implementar un sistema inmutable sobre las capas *Data* e *Index*, ya que el coste de consultar la capa *Online* tiende a ser inferior al de acceder a la capas de datos inferiores. El único coste adicional pasa por el proceso de *Merge* que, sin embargo, se integra en el funcionamiento habitual del sistema y se ejecuta de forma transparente al cliente.

La dinámica que implementa SOLID se inspira, parcialmente, en la arquitectura Lambda [14] diseñada para el procesamiento de *Big Data*, en un contexto genérico. Sin embargo, la especialización de SOLID en la gestión de datos semánticos nos permite optimizar la lógica de nuestras capas, diferenciando algunos puntos fundamentales respecto a lo propuesto en Lambda. Por comparación, Lambda es una arquitectura en tres capas: *Batch, Serving* y *Service*, en la que las operaciones de consulta se resuelven utilizando vistas precomputadas. Esto restringe la posibles consultas que se pueden hacer en un momento determinado ya que sólo podrán resolverse aquellas consultas con vistas asociadas. La capa *Batch* aborda el almacenamiento de *Big Data* sobre el mismo principio de inmutabilidad e implementa los mecanismos para la construcción de diferentes tipos de vistas sobre los datos. La capa *Serving* carga estas vistas para su consulta, mientras que la capa *Service* gestiona las necesidades del tiempo real mediante procesos de *actualización incremental* de los datos que almacena y de las vistas que construye sobre ellos. Esta capa se responsabiliza de recoger los datos generados en tiempo real y almacenarlos temporalmente hasta que se integren en el *Big Data* residente en la capa *Batch*. Una vez se decide esta integración, se recomputarían desde cero todas las vistas de la capa *Serving* y se eliminarían todos los datos en *Service*.

Las tres capas de Lambda se corresponden, en cierto modo, con las capas *Data*, *Index* y *Online*. Sin embargo, en Lambda es la propia capa de datos la encargada de realizar el proceso *batch* de integración, de ahí su nombre. Además, la gestión de consulta basada en vistas restringe qué consultas se pueden realizar en un momento dado, mientras que SOLID facilita la resolución de cualquier consulta expresada en SPARQL.

4. Implementación de SOLID

La sección actual plantea una propuesta de implementación de la arquitectura SOLID considerando su uso en un escenario general. No obstante, se podrían definir otras

posibles implementaciones para otros escenarios particulares. En primer lugar se describen las decisiones adoptadas sobre las tres capas de datos dirigidas, principalmente, por la adopción de RDF/HDT como formato básico para la representación de RDF. A continuación se plantea una propuesta de implementación de las capas *Merge* y *Service* que aprovecha las propiedades de RDF/HDT para mejorar su rendimiento efectivo.

4.1. Capas de datos

La capa **Data** garantiza la inmutabilidad del *Big Semantic Data* al utilizar RDF/HDT para su almacenamiento. Al mismo tiempo, al elegir este formato de serialización se garantiza el objetivo de mantenerla lo más compacta posible. Adicionalmente, la configuración de estructuras de datos compactas que propone RDF/HDT le da un valor añadido a la capa actual al posibilitar el acceso eficiente a los datos sin necesidad de descomprimirlos. Por lo tanto, esta capa asume la complejidad de gestionar el volumen del *Big Semantic Data* y establece la base sobre la que se implementa la capa *Index*.

La capa **Index** aprovecha la implementación de *Data* para construir un sistema de indexación ligero basado en HDT-FoQ [13]. Esta decisión no sólo mantiene el objetivo de trabajar en espacio comprimido (y con ello una gestión escalable del volumen del *Big Semantic Data*), si no que dota a SOLID de un rendimiento en consulta comparable al de las técnicas dominantes en la resolución de SPARQL [13], obteniendo con ello una gestión escalable de la velocidad de consulta.

La capa **Online** asume por completo la complejidad de la gestión de datos en tiempo real, desacoplando su implementación de la de las dos capas inferiores. Esta decisión arquitectónica es la que nos permite implementar las capas *Data* e *Index* utilizando RDF/HDT como HDT-FoQ, ya que sus respectivas representaciones son de *sólo lectura*. La implementación de esta capa está fuertemente ligada a la naturaleza específica del sistema de información diseñado sobre SOLID. En la sección siguiente se muestran varias opciones de implementación sobre tecnología NoSQL específicamente diseñada para el almacenamiento y consulta de RDF, dado que este tipo de soluciones han mostrado un rendimiento superior a las relacionales en lo relativo a la resolución de SPARQL [1]. Aún así, una implementación basada en tecnología relacional tendría su oportunidad siempre que garantizase una mejor tasa de inserción de datos.

4.2. Capas de procesamiento intensivo de datos

La capa **Merge** se implementa sobre un procesamiento en *batch* que implementa la integración de dos colecciones RDF serializadas en RDF/HDT, por lo cual los datos de la capa *Online* deberán transformarse previamente a esta representación si es que el *dump* se genera en algún otro formato. Una vez disponibles las dos representaciones, el algoritmo de integración las recorre en paralelo aprovechando que RDF/HDT almacena las ternas RDF de forma ordenada. En esta primera pasada se obtiene la información necesaria para generar el nuevo RDF/HDT y en una segunda pasada sobre las dos colecciones se serializa el nuevo *Big Semantic Data*. Una vez completado el proceso, se actualiza la configuración de la capa *Index* (de acuerdo a lo propuesto en [13]), se cargan las estructuras resultantes y, en paralelo, se eliminan de la capa *Online* los datos ya

integrados en el *Big Semantic Data*. A partir de este momento, el nuevo *Big Semantic Data* reemplaza al anterior, que es descartado.

La capa **Service** aprovecha la expresividad de SPARQL para ofrecer un espacio de consulta abierto ante cualquier necesidad potencial del cliente. Su propuesta de implementación sigue las directrices originales y desarrolla el pipeline básico. En la práctica, el filtro de integración de los resultados de la consulta recibirá primero los procedentes de la capa *Online*, facilitando el procesamiento de sus resultados antes de recibir los procedentes de la capa *Index*. Este hecho permite optimizar la implementación del filtro, por ejemplo, construyendo un índice asociativo sobre los resultados de la capa *Online* que mejore la eficiencia al procesar los obtenidos en la capa *Index*.

5. Resultados Experimentales

La sección actual muestra los primeros resultados experimentales de la arquitectura SOLID de acuerdo a su estado actual de desarrollo. Los objetivos principales de estos experimentos son i) estudiar el rendimiento de las capas de datos basadas en RDF/HDT respecto al que se obtendría con otras tecnologías comparables, ii) analizar el comportamiento de diferentes tecnologías NoSQL para afrontar la gestión de RDF en tiempo real y iii) evaluar la escalabilidad de la capa *Merge* con el crecimiento del tamaño del *Big Semantic Data*. En estos experimentos se deja al margen la capa *Service* dado que su optimización depende de cómo se implementen las capas *Online* e *Index*.

Consideraciones Iniciales. Todos nuestros experimentos se llevan a cabo utilizando el conjunto de datos *Linked Sensor Data*[10] (LSD). Esta colección contiene información recogida por 20000 estaciones meteorológicas situadas en distintos puntos de Estados Unidos. No obstante, el patrón que sigue la recogida y gestión de los datos, en LSD, es algo diferente al planteado en nuestro ejemplo ya que la operativa en tiempo real se lleva a cabo, sólo, durante los días que dura un fenómeno atmosférico.

La colección LSD describe datos sobre siete huracanes y una gran tormenta. Estos datos comprenden mediciones de temperatura, dirección y velocidad del viento, visibilidad, presión atmosférica o humedad entre otros, así como el *timestamp* del momento en que se tomaron. Por ejemplo, las mediciones recogidas durante los 8 días del huracán Katrina suponen un total de 200 millones de ternas RDF.

Para simular el análisis del rendimiento en tiempo real definimos un periodo continuo que suma la duración de todos los fenómenos atmosféricos. El punto de partida del estudio considera que la capa *Data* almacena un *Big Semantic Data* (denominado *Base*) con la descripción de todos los sensores en el estudio y las mediciones de la tormenta *Nevada* y los huracanes *Charley, Katrina* y *Wilma*. Por lo tanto, el primer día sobre el que estudiamos el comportamiento en tiempo real es el 10/09/2008, el comienzo del huracán Ike. Los datos recogidos en este día (referidos como *Día*) serán los utilizados para analizar el rendimiento de la capa *Online*.

Entorno de experimentación. La experimentación se ha realizado sobre una máquina Intel Core 2 Duo P7350 2Ghz 8Gb Ram DDR3. OSX 10.7.5 (Disco duro 160Gb 5400RPM). Las técnicas consideradas en el experimento son RDF/HDT (v.1.0), RDF-3x (v.0.3.5) y Apache Jena TDB (v.2.10.0, tdbloader2 con sort de coreutils 8.13).

[10] `http://wiki.knoesis.org/index.php/SSW_Datasets`

	# Ternas	Tamaño (GB) *-NTriples-*	Tamaño Datos (GB)			Tiempo de Generación (minutos)		
			RDF/HDT	RDF3X	Jena TDB	RDF/HDT	RDF3X	Jena TDB
Base	454 M	91,3	**4,8**	25	44	**112**	480	1320
Día	37,1 M	3,3	×	**1,9**	3,7	×	**12**	22

Cuadro 1. Tamaño de las representaciones y tiempos de generación.

Análisis del volumen. La columna *Tamaño Datos* del Cuadro 1 muestra los requisitos necesarios para almacenar los dos conjuntos de datos considerados. Por un lado, la fila Base representa el coste en GB que supone representar las capas *Data* e *Index* utilizando cada una de las tecnologías en estudio. Como puede observarse, el *Big Semantic Data* contiene 454 millones de ternas RDF que ocupan 91,3 GB serializadas en NTriples. La representación basada en RDF/HDT es claramente la más compacta de todas: apenas requiere 4,8 GB para almacenar los datos y los índices sobre ellos. Este resultado significa que la implementación de las dos capas utilizando RDF/HDT + HDT-FoQ ocupa *19 veces menos* que el contenido RDF directamente serializado en NTriples. Por su parte, RDF3X requiere 25 GB y Jena TDB 44 GB, lo que muestra la enorme ventaja en espacio que se consigue al utilizar nuestra propuesta.

Por su parte el análisis de la fila Día da una idea de la complejidad espacial que supone manejar los datos recogidos en tiempo real. Este subconjunto contiene 37,1 millones de ternas RDF que ocupan 3,3 GB en NTriples. La comparación en este caso es entre RDF3X y Jena TDB, ya que RDF/HDT ofrece una representación de "sólo lectura" cuyo uso no aplica a la capa actual. RDF3X requiere 1,9 GB para la capa *Online* mientras que Jena TDB usa, aproximadamente, el doble de espacio (3,7 GB).

Análisis de la velocidad de procesamiento. La columna *Tiempo de Generación* del Cuadro 1 muestra el tiempo necesario para obtener las representaciones de cada uno de los conjuntos de datos. Para el caso de la colección Base es un valor informativo, ya que el coste de generar las representaciones iniciales de las capas *Data* e *Index* se paga sólo al poner en funcionamiento el sistema. Como puede observarse, RDF/HDT tarda 127 minutos en obtener la configuración de ambas capas, mientras que RDF3X y JenaTDB requieren, respectivamente, 8 y 22 horas. El tiempo de que toma obtener la representación de Día es más significativo ya que plantea una estimación de la velocidad de escritura de la capa *Online*. Esto es, RDF3X tarda 12 minutos en representar la colección Día que contiene 37,1 millones de ternas, lo que supone una velocidad de procesamiento superior a los *3 millones de ternas por minuto*. Por su parte, JenaTDB reporta una velocidad inferior: $\approx 1,7$ millones de ternas por minuto.

Análisis de la velocidad de consulta. La tabla izquierda de la figura 2 resume los tiempos al consultar el *Big Semantic Data* de las capas inferiores y la colección en tiempo real de la capa *Online*. Las consultas utilizadas para este experimento han sido generadas de forma aleatoria y están disponibles en nuestro sitio web[11]. Los datos relativos a la colección Base se promedian sobre un conjunto de 1000 consultas SPARQL. RDF/HDT supera notablemente a las otras dos alternativas, promediando un tiempo de *0,036 segundos/consulta* respecto a los $0,761$ de RDF3X y $1,011$ de Jena TDB. Esta comparación muestra que RDF/HDT ofrece una velocidad de consulta entre 20 y 30 veces superior a la de las otras técnicas. La comparación en la capa *Online* favorece

[11] http://dataweb.infor.uva.es/weather_queries.tar.gz

	RDF/HDT	RDF3X	Jena TDB
Base	**0,036s**	0,761s	1,011s
Día	×	**0,007s**	0,011s

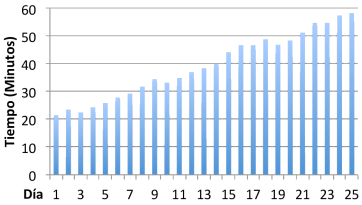

Figura 2. Tiempos medios de *consulta* (izquierda) y tiempos de *merge* (derecha).

a RDF3X, que requiere una media de *0,007 segundos/consulta* frente a los $0,011$ que necesita Jena TDB para resolver cada una de las 80000 consultas consideradas.

Con los números actuales, podemos concluir que RDF/HDT es la técnica más compacta y, a la vez, la más eficiente para implementar las capas *Data* e *Index*, superando a las otras dos técnicas consideradas. RDF3X se postula como la solución más eficiente para la capa *Online*, atendiendo tanto a su mayor rendimiento en las escrituras como en la resolución de consultas. Nótese también que el tiempo promedio de consulta reportado por RDF3X para *Día* es de $0,007$ segundos respecto a los $0,035$ que requiere RDF/HDT para *Base*, lo cual constata que el la complejidad de la consulta estará determinada, principalmente, por el acceso al *Big Semantic Data*.

Análisis del coste de integración de los datos. Finalmente, la Figura 2 (derecha) analiza el rendimiento de la capa *Merge* en un periodo temporal. La figura comprende los tiempos de *Merge* durante 25 días sucesivos en los que, diariamente, se integran en el *Big Semantic Data* los datos recogidos en dicho día. Por lo tanto, el tamaño del *Big Semantic Data* crece progresivamente hasta alcanzar un tamaño final de 1348 millones de ternas RDF, mientras que el tamaño de los datos diarios tiene un tamaño promedio de 32 millones. Como puede observarse, el tiempo de integración presenta un crecimiento lineal desde 20 a ≈60 minutos, lo que resulta un tiempo asumible en la práctica. Sin embargo, la tendencia lineal en el crecimiento de este valor supone un aspecto a optimizar sobre el algoritmo de integración actual como se indica en el trabajo futuro.

6. Conclusiones y Trabajo Futuro

En este artículo hemos presentado los principios arquitectónicos fundamentales para la gestión de *Big Semantic Data* en tiempo real. SOLID se conceptualiza como una arquitectura multicapa basada en el aislamiento de las complejidades subyacentes al consumo de grandes volúmenes de datos y su gestión en tiempo real. La escalabilidad en el almacenamiento y consulta de *Big Semantic Data* se ha delegado en las posibilidades que abre el uso de RDF/HDT para la serialización e indexación de RDF. Por otro lado, SOLID almacena y expone, para consulta, los datos generados en tiempo real utilizando tecnología NoSQL. Los resultados experimentales que hemos presentado, para cada una de las tecnologías con las que implementamos estas capas de datos, plantean un *baseline* consistente para el desarrollo de nuestro trabajo futuro.

La independencia planteada en la representación de la colección hace necesario el uso de sendos mecanismos de interrelación entre los datos obtenidos en tiempo real y aquellos que conforman el *Big Semantic Data*. El rendimiento mostrado, actualmente,

por la capa *Merge* nos conduce a buscar una alternativa más optimizada para la integración de los datos procedentes de la capa *Online*. Aparte del rediseño del propio algoritmo, estamos trabajando en una implementación basada en Map-Reduce con la que obtener una distribución de los costes y mejorar el impacto del proceso en el rendimiento global de la arquitectura. Por otra parte, también trabajamos en la optimización de la capa *Service* de acuerdo a los resultados experimentales obtenidos en este trabajo.

Finalmente, estamos avanzando en el desarrollo de una implementación ligera de SOLID destinada a su utilización en dispositivos móviles. Los requisitos, en este caso, plantean algunas particularidades respecto al escenario general (por ejemplo, se descarta la implementación de la capa *Merge*), aunque el ahorro espacial cobra mayor interés debido a la limitación computacional de este tipo de dispositivos.

Referencias

1. D. Abadi, A. Marcus, S. Madden, and K. Hollenbach. Scalable semantic Web data management using vertical partitioning. In *Proc. of VLDB*, pages 411–422, 2007.
2. D. Beckett, editor. *RDF/XML Syntax Specification*. W3C Recommendation, 2004.
3. E. Begoli and J. Horey. Design Principles for Effective Knowledge Discovery from Big Data. In *Proc. of Joint WICSA/ECSA*, pages 215–218, 2012.
4. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 2001.
5. S. De, T. Elsaleh, P. Barnaghi, and S. Meissner. An Internet of Things Platform for Real-World and Digital Objects. *Scalable Computing: Practice and Experience*, 13(1), 2012.
6. J. Fernández, M. Arias, M. Martínez-Prieto, and C. Gutiérrez. Management of big semantic data. In R. Akerkar, editor, *Big Data Computing*, chapter 4. Taylor and Francis/CRC, 2013.
7. J. Fernández, M. Martínez-Prieto, C. Gutiérrez, and A. Polleres. *Binary RDF Representation for Publication and Exchange (HDT)*. W3C Member Submission, 2011.
8. J. Fernández, M. Martínez-Prieto, C. Gutiérrez, A. Polleres, and M. Arias. Binary RDF representation for publication and exchange. *Journal of Web Semantics*, 19:22–41, 2013.
9. Y. Genovese and S. Prentice. Pattern-Based Strategy: Getting Value from Big Data. Gartner Special Report, June 2011.
10. T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011. `Available at http://linkeddatabook.com/`.
11. M. Loukides. Data Science and Data Tools. In *Big Data Now*, chapter 1. O'Reilly, 2012.
12. F. Manola and E. Miller, editors. *RDF Primer*. W3C Recommendation, 2004.
13. M. Martínez-Prieto, M. Arias, and J. Fernández. Exchange and Consumption of Huge RDF Data. In *Proc. of ESWC*, pages 437–452, 2012.
14. N. Marz and J. Warren. *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. Manning, 2013.
15. G. Navarro and V. Mäkinen. Compressed full-text indexes. *ACM Computing Surveys*, 39(1):article 2, 2007.
16. T. Neumann and G. Weikum. The RDF-3X engine for scalable management of RDF data. *The VLDB Journal*, 19:91–113, 2010.
17. D. Palmisano. Introduction to Linked Data. Cambridge Semantics: Semantic University, `http://bit.ly/ZC8OKu` (consultado 05-04-2013).
18. E. Prud'hommeaux and A. Seaborne, editors. *SPARQL Query Language for RDF*. W3C Recommendation, 2008.
19. R. Styles. *RDF, Big Data and The Semantic Web*. `http://dynamicorange.com/2012/04/24/rdf-big-data-and-the-semantic-web/` (consultado 05-04-2013).
20. G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 25(3):38–49, Mar. 1992.