Descripción de pruebas de benchmark para plataformas de tercera generación

Martín de la Rubia L., Algorri M., Zorrilla M. y Drake J.M.,

[1] Grupo de Ingeniería Software y Tiempo Real (ISTR), Universidad de Cantabria
[www.istr.unican.es]

Resumen. La irrupción del big data y la computación en la nube ha impulsado un cambio de paradigma en la construcción de nuevos sistemas basados en plataformas distribuidas escalables y orientadas al dato como servicio. La existencia de diversas tecnologías y la necesidad de evaluar el rendimiento de las aplicaciones construidas con ellas, tanto en fase de prototipo como ya una vez implementadas y desplegadas en el entorno operativo, nos ha llevado a proponer un modelo de datos para describir pruebas de benchmark destinadas a la comparación de estas plataformas de tercera generación. El modelo incorpora información sobre todos los aspectos de la prueba: recursos, fuentes de datos, cargas de trabajo y métricas; cubre varios casos de uso y permite adaptar la información que contiene a las diferentes fases del ciclo de desarrollo del sistema. En las fases iniciales de desarrollo de prototipos, el modelo describe estimaciones de la carga de trabajo, de las prestaciones previstas para los recursos y componentes del sistema y de las métricas que se quieren valorar; mientras que en las fases finales de validación, el modelo sólo ha de incluir la identificación de las fuentes que generan las cargas de trabajo, de los recursos utilizados y de los componente desplegados, a fin de evaluar las métricas de interés.

Keywords: Big data, distributed system, benchmarks and performance metrics.

1 Plataformas de tercera generación: modelado y evaluación

La implantación de la *cloud computing* y el *big data* ha dado lugar a la aparición de las denominadas plataformas de tercera generación [1]. Estas se caracterizan por ser arquitecturas distribuidas y escalables que dinámicamente se dimensionan en base al volumen de computación que se les requiere. Surgen para dar respuesta a:

1) La necesidad de disponer de alta capacidad de computación en cualquier entorno fijo o móvil (*Ubiquitous* & *Mobile Computing*),
2) La capacidad de gestionar información masiva en tiempo real (*Big Data*), sin necesidad de almacenarla para ser procesada en diferido.
3) La posibilidad de escalar dinámicamente la capacidad de procesamiento en base a su virtualización y deslocalización (*Cloud & Fog Computing*).
4) La interacción dinámica de las aplicaciones con entornos inteligentes (*IE* & *IoT*) y redes sociales (*Social Network*).

5) La generalización del uso de los nuevos paradigmas de computación basados en la inteligencia artificial (*AI*) y el aprendizaje automático (*Machine Learning*).

Aunque existen muchas opciones para implementar las plataformas de tercera generación, nuestro trabajo se centra en plataformas compuestas por conjuntos de nodos de procesamiento físico o virtual, redes de comunicación y software de intermediación, gestionados bajo dos paradigmas:

- **DaaS** (*Data as a Service*): El dato constituye el nivel más alto de la arquitectura software. Los tipos de datos (*topics*) no sólo caracterizan la información que representan, sino que también definen los requisitos no funcionales con que deben ser gestionados (persistencia, durabilidad, disponibilidad, seguridad, integridad, etc.) y en muchos casos, se definen antes que las aplicaciones a fin de potenciar el desarrollo inter-empresarial de componentes. Por otra parte, las tareas de procesamiento son componentes software desacoplados que se ejecutan en los nodos de la plataforma y que interaccionan entre sí actuando como publicadores o subscriptores de los tipos de datos definidos por la aplicación y que representan el flujo de control interno dentro de sus transacciones. Las aplicaciones se conciben como conjuntos de tareas de procesamiento que se ejecutan en base a la ocurrencia de determinados patrones en los datos generados por el entorno o por otras tareas de la misma aplicación.
- **PaaS** (*Platform as a Service*): Las plataformas están constituidas por conjuntos de recursos computacionales físicos disponibles en el propio entorno (*fog*) y/o recursos computacionales virtuales externos (*cloud*) que pueden ser reclutados dinámicamente en función de la carga de trabajo que se genere.

Nuestro grupo, dentro del proyecto del plan Nacional “Sistemas Informáticos Predecibles y confiables para la Industria 4.0” (TIN2017-86520-C3-3_R) está desarrollando la plataforma de tercera generación *P3forI4* para dar soporte a la Industria 4.0 y que se ha implementado utilizando tecnologías del proyecto de software libre Apache [2]: AVRO [3], ZooKeeper [4], Kafka[5] y Spark [6]. Por último, MemSQL[7] se utiliza para persistir en memoria la información que lo requiera.

En la fase de diseño de la plataforma se ha necesitado comparar las tecnologías alternativas disponibles, y también, evaluar el rendimiento de la plataforma en la ejecución de aplicaciones que son propias en la Industria 4.0. Con este objetivo se han tenido que aplicar diferentes benchmark de evaluación, unos tomados de las referencias bibliográficas y otros de casos de uso propuestos por las empresas que participan en el proyecto. La aplicación manual de benchmark es una tarea tediosa, que requiere adaptar las fuentes de los datos y los componentes de procesamiento a las interfaces de la plataforma, suplir con emuladores los elementos que no están disponibles e incorporar y gestionar la instrumentación y los monitores con los que se valoran las pruebas. Esta experiencia, es el origen de este trabajo. Se consideró que la aplicación de benchmarks es una actividad que puede ser asistida mediante herramientas, y con ellas, se reduce el esfuerzo de su aplicación, se incrementa la reproducibilidad y se sistematiza su análisis y documentación. Como paso previo al desarrollo y aplicación de herramientas, se ha considerado necesario proponer un modelo de datos que des-

criba las pruebas de benchmark con independencia de las tecnologías de las plataformas y de la naturaleza de las propias pruebas.

El modelo incorpora información sobre todos los aspectos de la prueba: de la plataforma en la que se ejecuta, de las fuentes que generan los flujos de datos del entorno, de las tareas de procesamiento que ejecuta y de las métricas que deben ser evaluadas para caracterizar la ejecución. El modelo cubre varios casos de uso de las pruebas de benchmark: la comparación de las prestaciones de diferentes plataformas en la ejecución de una misma aplicación; el efecto sobre la ejecución de la aplicación del escalado y configuración de la plataforma; la repercusión de la ejecución de la aplicación sobre otras aplicaciones que también se están ejecutando en la plataforma; y la evaluación de la eficiencia de la descomposición en componentes de la aplicación para su ejecución sobre la plataforma. El modelo además permite adaptar la información que contiene a las diferentes fases del ciclo de desarrollo del sistema. En las fases iniciales basadas en prototipos, el modelo describe estimaciones de la carga de trabajo, de las prestaciones previstas para los recursos y componentes del sistema y de las métricas que se quieren valorar. Mientras que, en las fases finales de validación, con el sistema ya implementado, el modelo sólo ha de incluir la identificación de las fuentes que generan las cargas de trabajo, de los recursos utilizados y de los componentes desplegados, a fin de evaluar las métricas de interés.

La organización de este documento es como sigue. En la sección 2 se revisan diferentes tipos de benchmark propuestos en la bibliografía y se analizan la forma en que se formulan y las diferencias que existen con nuestra propuesta. En la Sección 3 se describe el modelo de datos que se propone. En la Sección 4 se plantea la información complementaria que se requiere para adaptar la descripción de la prueba a cada tecnología, y en particular, a las tecnologías con las se ha trabajado. En la Sección 5, se presenta un ejemplo de descripción de un benchmark y su uso en las diferentes fases del desarrollo de una aplicación. Por último, en la Sección 6 se resumen los resultados del trabajo y las líneas abiertas para cubrir los objetivos propuestos.

2 Formulación y aplicación de benchmark en la bibliografía.

Desde hace más de 30 años, la industria se ha preocupado por desarrollar benchmarks estandarizados para comparar sistemas e implementaciones de forma objetiva, persiguiendo además que fueran sencillos, portables y escalables [8]. Destaca el trabajo realizado por organizaciones consolidadas como Transaction Processing Performance Council (TPC) centrada en medir el rendimiento de bases de datos para evaluar sistemas transaccionales (TPC-C, TPC-E) y decisionales (TPC-H, TPC-DS) de forma estándar, objetiva y verificable [9] y la Standard Performance Evaluation Corporation (SPEC) focalizada en evaluar el rendimiento y la eficiencia energética para la última generación de sistemas informáticos.

Con el advenimiento de la *cloud computing*, han ido surgiendo entornos para evaluar el rendimiento de sistemas basados en MapReduce como SWIM [10], GridMix,

SparkBench[11], así como sistemas de bases de datos en la nube y bajo el paradigma NoSQL como YCSB[11], OLTP-Bench [12] o LinkBench[13]. Por ese tiempo, proveedores importantes con Cloudera e Intel a la cabeza, trataron de definir BigBench, an Industry Standard Benchmark for Big Data Analytics [14], tomando elementos de los esfuerzos de benchmarking existentes en el espacio Big Data (como YCSB, TPCxHS, GridMix, PigMix, HiBench, Big Data Benchmark y TPC-DS). En 2014, TPC debido a la importancia que estaban adquiriendo estos sistemas, publicó su TPCx-BB [15] con el fin de medir el rendimiento de sistemas Big Data basados en Hadoop, siguiendo su habitual forma de proceder, esto es, replicar sobre distintas configuraciones la ejecución de 30 consultas analíticas en un contexto de venta minorista utilizando Hive o Spark sobre datos estructurados y aplicando algoritmos de aprendizaje automático sobre datos semiestructurados y no estructurados especificados por el usuario. Otra iniciativa destacable es "Bigdatabench: A big data benchmark suite from internet services" [16] y que actualmente está disponible públicamente en la web su versión 4.0. Su enfoque es totalmente distinto, separa la especificación del benchmark de la implementación para así poder enfrentarse a la complejidad, diversidad y la rápida evolución de los diferentes tipos de cargas de trabajo que crean Deep Learning y Big data. Este se basa en el concepto de *dwarf*, una abstracción de unidades de computación de cada tipo de carga independientemente de su implementación que le evita implementar cada tarea específica, que además de costoso y no escalable, plantea dificultades en la reproducibilidad e interpretación de los datos de rendimiento. Este, no trabaja sobre un único modelo de datos sino que ofrece diferentes generadores de datos basados en casos reales lo que le permite modelar diferentes escenarios. Proporciona trece conjuntos de datos del mundo real (bioinformática, comercio electrónico, redes sociales, motores de búsqueda, etc.) y cuarenta y siete benchmarks que cubren siete tipos de cargas de trabajo (servicios en línea, análisis fuera de línea, análisis de gráficos, inteligencia artificial, almacenes de datos, NoSQL y streaming). Existen otras iniciativas de propósito más específico como AdBench[17], específico para compañías de marketing o IoTbench centrado en el contexto IoT[18].

En su mayoría, los benchmarks existentes permiten comparar los rendimientos de diferentes sistemas en condiciones experimentales dadas, lo que ayuda a los proveedores a posicionar sus productos en relación a sus competidores, y a los usuarios seleccionar soluciones estratégicas basadas en información objetiva. Pero estos presentan modelos de datos y cargas de trabajo fijas. Estos benchmarks, por tanto, tienen su cuota de utilidad pero en un escenario tan creciente y cambiante, carecen de la posibilidad de personalizar los mismos con las fuentes de datos y procesos reales de un entorno que ya existe o simular uno específico o integrar ambas posibilidades para emular el efecto que tendría un nuevo proceso sobre la plataforma existente. Este es el objetivo de nuestra propuesta, diseñar una estrategia que se adapte a las distintas fases de desarrollo desde los prototipos hasta la de validación.

3 Modelos de datos para la descripción de pruebas de benchmark y casos de usos.

El modelo de datos para la descripción de una prueba de benchmark organiza estructuradamente toda la información que se requiere para satisfacer los casos de uso que se contemplan:

- **Comparación de tecnologías**: comparar las prestaciones y eficiencia de diferentes plataformas, cada una de ellas implementadas con una tecnología diferente.
- **Evaluación de escalado y configuración de la plataforma**: evaluar el dimensionado de los recursos que constituyen la plataforma y la configuración de los recursos de la misma.
- **Diseño modular de las aplicaciones**: evaluar las alternativas del diseño de componentes de las aplicaciones en lo que concierne al uso eficiente de los recursos frente al escalado de la plataforma.
- **Valoración del efecto sobre el sistema del despliegue de una aplicación**: valorar el efecto que sobre un entorno en operación produce el despliegue y ejecución de nuevas aplicaciones.

La descripción de la prueba debe adaptarse al estado de implementación del sistema y a la disponibilidad o no del entorno operativo real. Cuando el código de la aplicación y el entorno operativo están disponibles, la descripción del benchmark sólo identifica los elementos que participan y describe las métricas que deben ser evaluadas. Por el contrario, si la prueba se formula en fase de prototipo en la que el entorno o el código de los componentes de procesamiento no están disponibles, la descripción de benchmark debe incluir la información para que durante su ejecución puedan ser emulados aquellos componentes que no están disponibles.

La Figura 1, muestra los tipos de datos de más alto nivel del modelo que describe una prueba de benchmark. La clase raíz P3BM_Proof identifica la prueba y constituye un contenedor plano (ownedElement) de todos los elementos del modelo (*P3BM_Element*). La clase abstracta *P3BM_NamedElement* se introduce para establecer que todos los elementos del modelo tengan nombre e identificador (clave). Un modelo puede referenciar a otros modelos (referencedProof), y a través de ello, se facilita la organización del modelo con secciones reutilizables e intercambiables entre pruebas.

Las clases *P3BM_Resource*, P3BM_EnvironmentStreamData, *P3BM_Workflow* y *P3BM_Metric* son los tipos de datos abstractos que describen los cuatro aspectos que intervienen en la descripción de una prueba de benchmark: las características de los recursos que constituyen la plataforma, lo flujos de datos que introduce el entorno y que constituyen la carga de trabajo, las tares de procesamiento que se ejecutan y las medidas que deben ser evaluadas en la ejecución de la prueba.

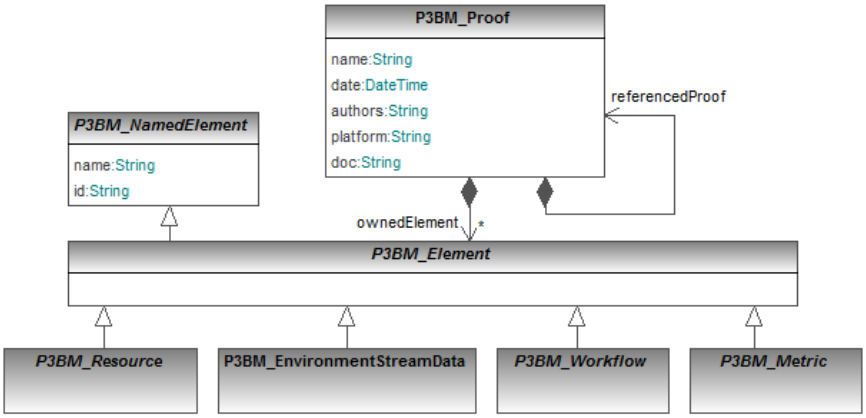

Fig. 1. Clases raíces del modelo de una prueba de benchmark.

Las Figura 2 muestra los tipos de datos genéricos que describen la información relativa a los diferentes tipos de recursos de la plataforma. P3BM_ProcessingNode caracterizan los nodos de procesamiento de la plataforma que participan en la prueba, a este nivel de abstracción incluye atributos que describen el número de procesadores de que dispone (concurencyLevel), la cantidad de memoria (memorySyze_bytes) y su capacidad de procesamiento relativa (speedFactor). Se contemplan sus especializaciones como nodos físicos disponibles en el entorno (P3BM_PhysicalProcessingNode) o como recursos virtuales contratados en la nube (P3BM_VirtualProcessingNode). P3BM_Network caracteriza el efecto sobre la prueba de benchmark de la red de comunicaciones junto al servicio de comunicaciones que la gestiona.

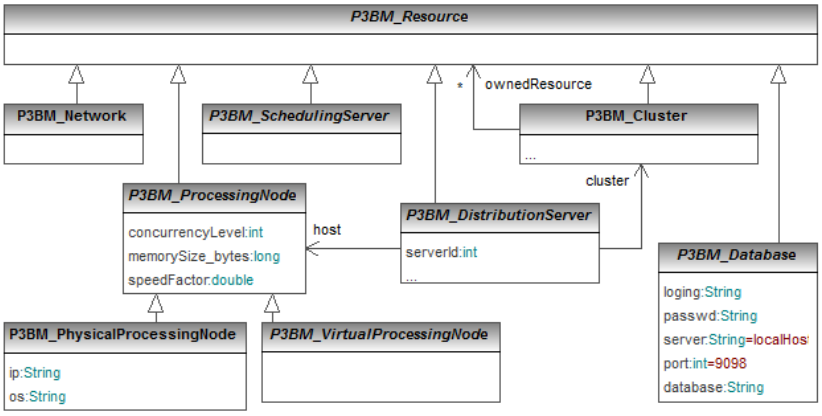

Fig. 2. Tipos de datos genéricos que describen los recursos de la plataforma.

P3BM_SchedulingServer identifica y caracteriza el planificador distribuido de tareas de la plataforma que gestiona la asignación de las tareas de procesamiento a los recursos de la plataforma. *P3BM_DistributionServer* describe la presencia de un recurso de gestión del sistema distribuido que facilita el acceso seguro a la información compartida desde todos los nudos, así como la replicación en segundo plano de la

información almacenada en los nodos para garantizar su integridad frente a caída de los mismos. *P3BM_Database* caracteriza los recursos de almacenamiento de datos utilizados en ciertas tareas que persisten la información. Por último, *P3BM_Cluster*, permite agrupar conjuntos de recursos presentes en la plataforma a fin de que puedan ser referenciados como una agrupación.

La Figura 3 muestra los tipos de datos que caracterizan a los flujos de datos de una prueba de benchmark. Cualquier tipo de flujo de datos tiene como raíz el tipo de datos abstracto *P3BM_StreamData* y tiene como atributos identificadores heredados de *P3BM_NamedElement*, una longitud de mensaje (messageSize_bytes) y un conjunto de campos específicos (fieldDescription) definidos con un identificador (clave) y un tipo de dato. Se consideran tres tipos de flujos de datos:

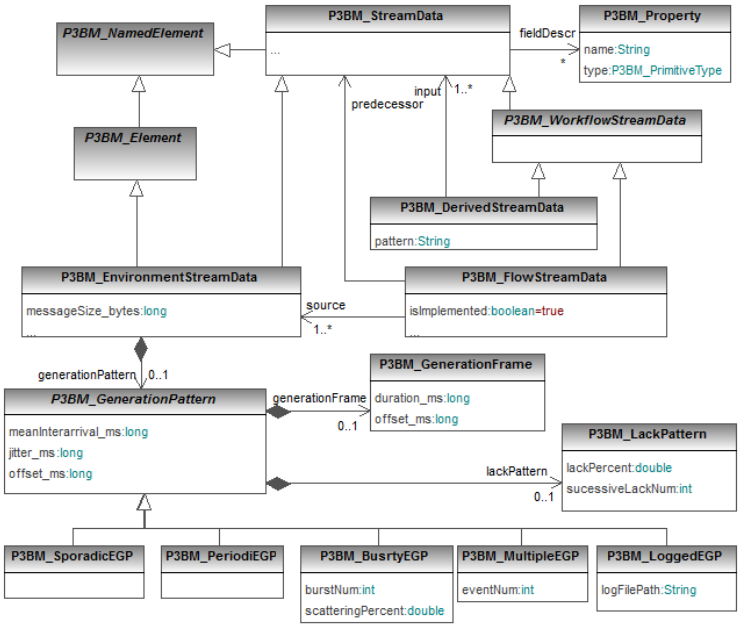

Fig. 3. Tipos de datos que describen un flujo de datos.

- P3BM_EnvironmentStreamData: Describe los flujos de datos que son generados por el entorno y que definen la carga de trabajo a la que está sometido el sistema en la prueba. Los flujos de datos de este tipo son declarados dentro de la lista ownedElement de la prueba. Si la prueba de benchmark se ejecuta sobre un entorno real y el flujo de eventos existe, la descripción sólo incluye su identificación, por el contrario, si la prueba se lleva a cabo en fase la fase de prototipo se le asocia una estructura de datos (*P3BM_GenerationPattern*), que proporciona la información necesaria para que su generación sea emulada por el gestor del benchmark. En la Figura 3 se muestras las alternativas de generación que se han definido.
- P3BM_FlowStreamData: Como se ha indicado en la Sección 1, la capacidad de escalar frente a incrementos de la carga de trabajo, resulta de replicar sus compo-

nentes con la multiplicidad que sea necesaria y desplegarlos por los recursos que disponga la plataforma en que se ejecutan. Este tipo de flujos de datos, representan el medio con el que se describe el flujo de control y de datos entre los componentes que ejecutan las tareas de las aplicaciones que se encuentran desplegados en cualquier nudo de la plataforma. Los flujos de datos de este tipo se declaran agregados (workflowEvent) dentro de los elementos *P3BM_Workflow* que describen las aplicaciones. Los atributos source y predecessor, se introducen con el fin de seguir las trazas del flujo de control de las aplicaciones y evaluar las latencias de las tareas y los tiempos de ejecución de las aplicaciones.

- P3BM_DerivedStreamData: Representa flujos de datos virtuales (sin realidad física en la plataforma) que resultan de aplicar una función de filtro a unos o varios flujos de datos que se generan en la prueba de benchmark. Un uso frecuente es la determinación de la condición de disparo de una aplicación frente a eventos del entorno o en la condición de disparo de una tarea de procesamiento dentro de una aplicación. Los dos atributos específicos del tipo de dato que describe estos eventos son la lista de flujos de datos que se referencian en la función de filtro (input), y la propia función de filtro (pattern) que se formula como una sentencia EPL [19] que es de tipo SQL pero con una cláusula RETAIN que restringe la consulta a ventanas temporales o por número de datos tanto de forma estática como deslizante.

En la Figura 4 se muestra el tipo de datos P3BM_Workflow que describe un flujo de procesamiento en una prueba de benchmark. Básicamente es un contenedor de las tareas de procesamiento (task) y de los eventos de control de flujo (workflowStreamDara) que representan el flujo de control y la transferencia de datos entre ellas. El atributo rootTask referencia la tarea que inicia un flujo de procesamiento. Todas las tareas que se ejecutan dentro de una misma ejecución del flujo de trabajo generan eventos de control de flujo que referencian como source al evento triggerEvent de la rootTask.

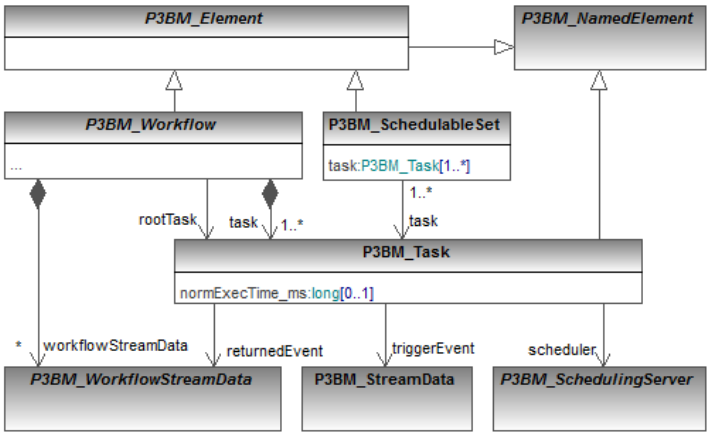

Fig. 4. Tipos de datos que describen un flujo de procesamiento.

Una tarea representa una actividad que es parte de un flujo de procesamiento y que ejecuta un componente software desplegable. En el modelo se caracteriza por la in-

formación incluida en el tipo de dato P3BM_Task. Incluye la referencia triggerEvent al flujo de datos (de entorno, de control de flujo o derivado) que inicia su ejecución, también la referencia returnedEvent al flujo de datos (de control de flujo) que genera cuando finaliza su ejecución. El atributo scheduler referencia el planificador distribuido que reparte la carga de procesamiento entre todas las réplicas de los componentes que ejecutan la tarea y que se han desplegado en la plataforma.

Cuando el benchmark se aplica en fase de prototipo el atributo opcional normalizedExecutionTime_ms representa el tiempo medio de ejecución de la tarea normalizado a un nodo de procesamiento de referencia (speedFactor=1.0). El entorno de gestión del benchmark, en base a esta información emula una tarea con el comportamiento especificado.

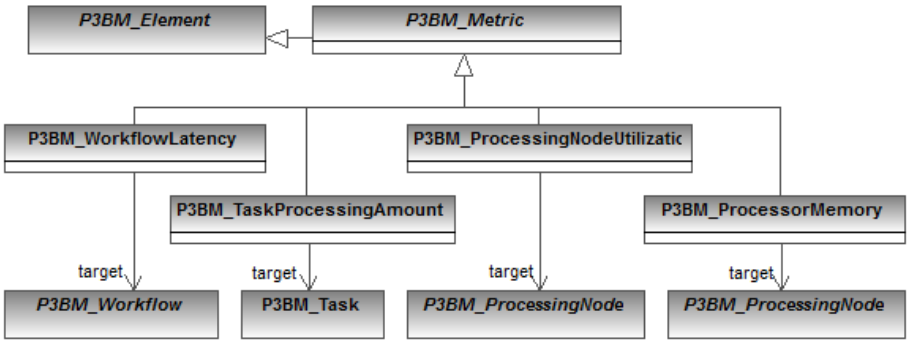

Fig. 5. Datos que describen métricas que pueden ser evaluadas en el benchmark.

En la Figura 5 se muestran los tipos de datos que describen medidas que se pueden evaluar durante la ejecución de la prueba de benchmark. Como lo habitual es tratar con código legado, no se basan en la instrumentación del código, sino en requerir las medidas de los servicios de monitorización que ofrecen los recursos o el middleware. Por ello, la información que incluye el modelo sólo especifica la naturaleza de la medida e identifica el elemento al que se aplica (target). Las métricas que pueden aplicarse a una prueba dependen de los objetivos del benchmark y pueden ser muy variadas. En la Figura 5 solo se han incluido cuatro métricas, relativas a las latencias de ejecución de los flujos de procesamiento (P3BM_WorkflowLatency) y de la cantidad de procesamiento de las tareas (P3BM_TaskProcessingAmount), a los niveles de utilización de los recursos (P3BM_ResourceUtilization) y a la cantidad de memoria principal que se usa en la prueba (P3BM_ProcessorMemory). Otras métricas sobre consumo de energía, fiabilidad, disponibilidad, etc. pueden ser incluidas si los servicios de monitorización de la plataforma las evalúan.

4 Personalización del modelo de datos a la tecnología de la plataforma.

El modelo propuesto en la Sección 3 es independiente de la tecnología utilizada. Sin embargo, aspectos como la configuración de los recursos y del middleware, o los

servicios de monitorización que dan soporte a la evaluación de las métricas son dependientes de la tecnología y requieren incluir nuevos atributos específicos. Esto se ha resuelto generando para cada tecnología nuevos tipos de datos personalizados.

En la Figura 6 se muestran las clases de datos especializadas incluidas para especificar pruebas de benchmark en plataforma P3forI4 basada en tecnología del proyecto Apache que se está desarrollando. Por ejemplo, el tipo de dato P*3BM_Scheduling Server*, que describe la estrategia de despliegue de las tareas de las cargas de trabajo en los recursos de procesamiento de la plataforma debe ser configurado en base a la tecnología del middleware utilizada. Para especificar la prueba de benchmark si se orienta a su ejecución en la plataforma P3forI4 se ha de implementar con los tipos de datos PABM_KafkaScheduler o PABM_SparkScheduler según el elemento en el que se haga caer la distribución de la carga de trabajo entre los recursos.

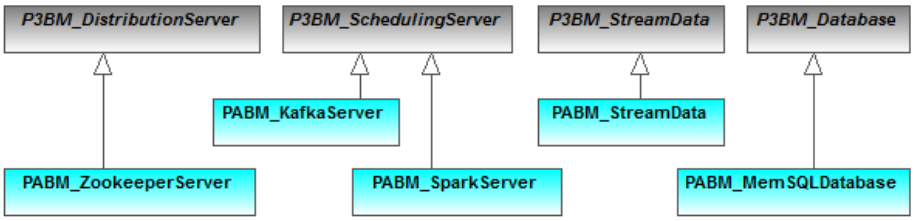

Fig. 6. Personalización de los tipos de datos para la plataforma P3forI4.

La personalización de la descripción de las pruebas en base a la tecnología supone un inconveniente cuando su objetivo es comparar tecnología para unas pruebas de referencia, ya que en este caso hay que generar una descripción de la prueba específica para cada tecnología que se compara.

5 Ejemplo de uso de una prueba de benchmark.

Como ejemplo de uso de las pruebas de benchmark, se presenta el caso de una empresa de distribución eléctrica, que aprovechando la información que transmiten los contadores inteligentes que ha implantado, analiza el consumo horario de cada uno de sus clientes. Cada contador transmite diariamente dos informaciones: una curva horaria de consumo (CH) que es poco fiable y la lectura real del contador (RM) que si es fiable. Ambas informaciones se reciben en la madrugada, y se almacenan en un registro para procesarla en diferido. Actualmente la información se procesa con un único mainframe que tarda aproximadamente 12 horas en procesar la información de sus 700.000 clientes. Como se muestra en la Figura 7, se almacenan las curvas horarias que han sido caracterizadas como válidas. De las restantes, lo que se almacenan es una estimación de ellas, obtenida en base al valor recibido y a valores registrados en la misma fecha en el año anterior para el mismo cliente. Posteriormente, cuando se recibe la RM de cada cliente se almacena, y además se revalúa de nuevo la CH, para que sus curvas horarias sean coherentes con el consumo.

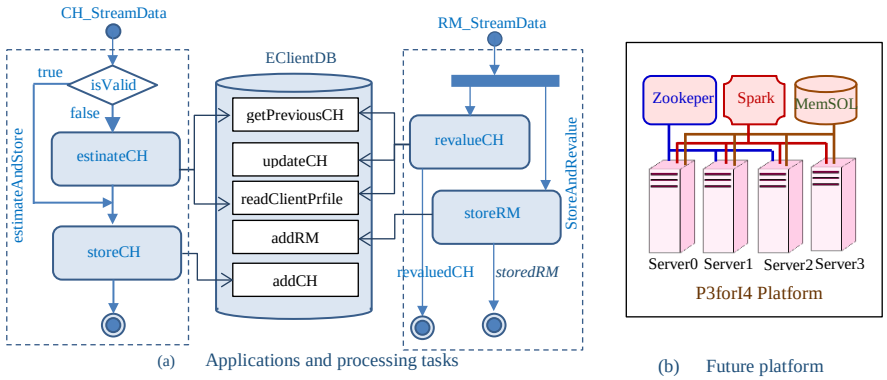

Fig. 7. Aplicaciones y futura plataforma del ejemplo.

Ante la previsión de tener que duplicar el número de clientes, la compañía eléctrica necesita escalar la plataforma para poder hacer frente a los nuevos flujos de datos. A tal fin, quiere evaluar el comportamiento de la plataforma que se muestra en la Figura 7.b., que está compuesta de 4 nodos de procesamientos con capacidad similar a la del mainframe en uso, utiliza un middleware de distribución Zookeeper que garantiza la integridad de los datos frente a fallos, planifica la ejecución en tiempo real utilizando Spark, y persiste los resultados utilizando una base de datos distribuida MemSQL.

La compañía utiliza los registros históricos de que dispone para valorar, a modo de benchmark, la futura plataforma. Para esta tarea es de gran ayuda disponer de una descripción formal de la prueba, tal como se ha descrito en este trabajo. Como se muestra en la Figura 8, la descripción de la prueba se compone de estructuras de datos que describen la plataforma, las fuentes de datos que proceden de los clientes, las tareas de procesamiento que se realizan y las métricas que deben ser evaluadas durante la ejecución.

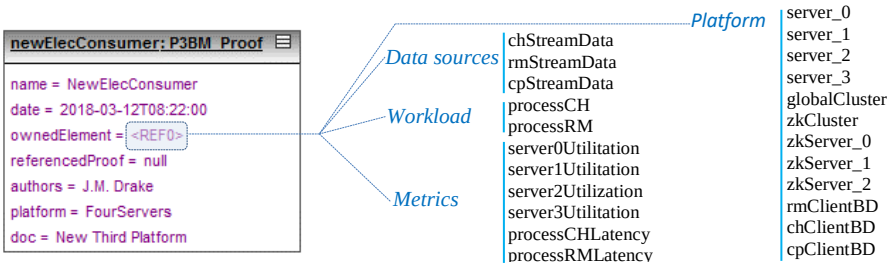

Fig. 8. Secciones y elementos de la descripción de la prueba del ejemplo.

En la Figura 9 se describe la plataforma. De cada nodo de procesamiento (server_?) se proporciona sus identificadores (name, id, ip, os), el número de procesadores (concurrencyLevel), la memoria de que disponen (memorySize_bytes) y su capacidad de procesamiento (speedFactor), que en este caso, es igual a la del mainframe en el que ha sido evaluada la carga de procesamiento de las tareas.

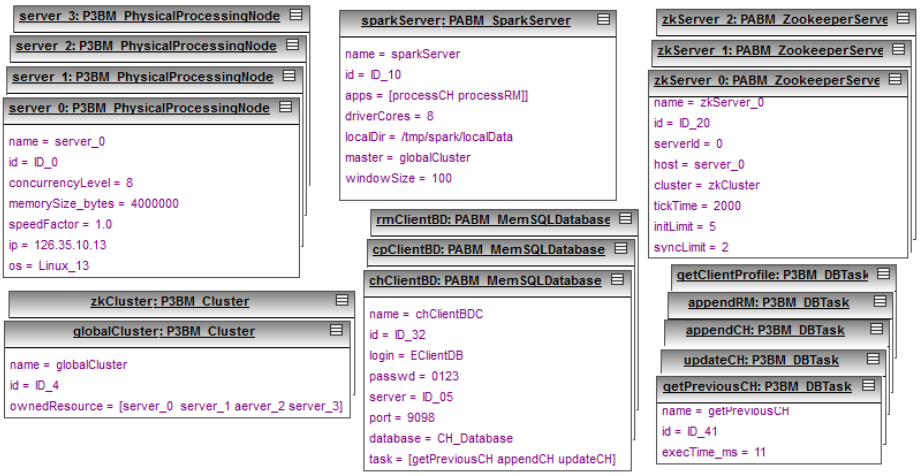

Fig. 9. Descripción de la futura plataforma de la prueba del ejemplo.

sparkServer describe el planificador de la ejecución de las tareas sobre la plataforma distribuida. Describe los nodos de la plataforma (master) en los que despliegan las tareas de las aplicaciones especificadas (apps), el número de procesadores que utiliza en cada nodo(driverCores), y la longitud en mensaje de las tandas que utiliza (windowSize). El gestor de distribución es el middleware Zookeeper (zkServer_?). Mantiene 3 réplicas de la información en los nudos de procesamiento del zkCluster (cluster), por lo que garantiza la integridad de los datos frente a la caída de un nodo. Sincroniza las réplicas cada 4 segundos (tickTime*syncLimit) y comprueba la operatividad de los nudos cada 10 segundos (tickTime*initLimit). Los resultados son persistidos en tres bases de datos de tipo MemSQL (??ClientBD), cuyos datos de acceso están incluidos (login, passwd, port, database), que se despliegan por los nodos del globalCluster (server_?) y sobre la que se realizan las consultas y/o actualizaciones que se describen (task).

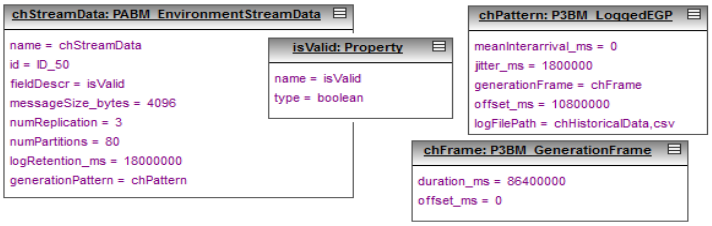

Fig. 10. Descripción del flujo de entrada chStreamData del ejemplo.

En la Figura 10 se muestra como ejemplo uno de los dos flujos de datos de entrada que se procesan. El flujo chStreamData se transmiten en mensajes de 4096 bytes (messageSize_bytes), se distribuyen en 80 particiones (partitions) para garantizar la máxima concurrencia, son replicados por triplicados (numReplication), se mantienen persistidos durante 5 horas (logRettention_ms) y tiene declarado el atributo isValid (fieldDescr). Se genera (generationPattern) con los datos del fichero chHistoricalData.csv, que se introducen durante 30 minutos a partir de las 3 horas de la mañana. Los datos se generan (generationFrame) con un marco de repetición diario (duration_ms).

En la Figura 11, se muestra como ejemplo la descripción de la aplicación processCH. Está compuesta por dos tareas estimateCh y storeCH y declara 4 flujos de datos (notValidCH, estimatedCH, validCh y storedCH) para sincronizar su ejecución. Por ejemplo, el flujo de datos notValidCh resulta de aplicar un filtro (pattern) al flujo de entrada (input) y estará compuesto por los datos del flujo chStreamData que tengan (isvalid=true). Este flujo se define como activador de la tarea estimateCH.

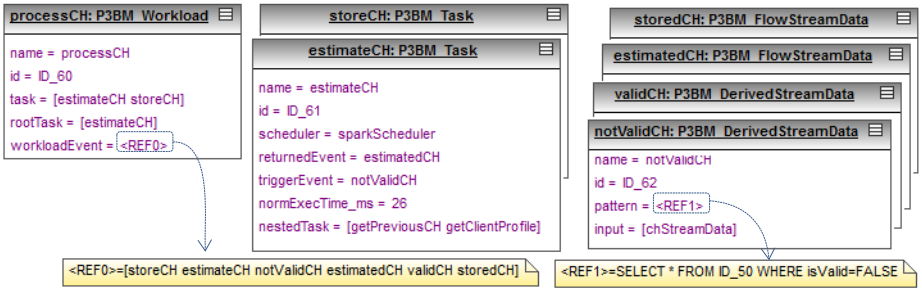

Fig. 11. Descripción de la aplicación processCH del ejemplo.

En la Figura 12 se describen las métricas que se aplican en la prueba. Estas son las utilizaciones de los cuatro nodos de procesamiento, y los tiempos de ejecución de las dos aplicaciones.

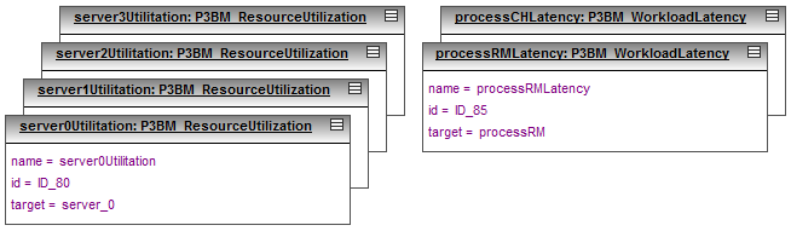

Fig. 12. Descripción de las métricas a aplicar en el ejemplo.

6 Resultados y líneas futuras.

Este trabajo presenta un modelo de datos para describir pruebas de benchmark que evalúen aplicaciones construidas sobre plataformas de tercera generación. Este artefacto software describe todos los elementos necesarios para definir estas pruebas tanto en la fase de prototipo, estimando las cargas de trabajo y las prestaciones de los recursos, como en la fase de validación en un entorno real en el que el modelo sólo ha de incluir la identificación de las fuentes que generan las cargas de trabajo, de los recursos utilizados y de los componentes desplegados, a fin de evaluar las métricas de interés. Así mismo se muestra un caso de aplicación en la Industria 4.0, sector al que va dirigido la plataforma de tercera generación que se está construyendo dentro del proyecto nacional TIN2017-86520-C3-3_R.

Actualmente se está diseñando la herramienta CASE que en base a este modelo permita al usuario final configurar las pruebas, ejecutarlas y analizar los resultados.

Agradecimientos

Este trabajo ha sido financiado por el Gobierno de España a través de los proyectos TIN2014-56158-C4-2-P y TIN2017-86520-C3-3_R.

References

1. Kwang K.: 'Third platform' shift triggers enterprise software evolution. ZDNet, «http://www.zdnet.com/article/third-platform-shift-triggers-enterprise-software-evolution/» (2013).
2. The Apache Software Foundation «http://www.apache.org/» (2017).
3. The Apache Avro project: a data serialization system. «http://avro.apache.org/».
4. Junqueira F. y Reed B.: ZooKeeper: Distributed process Coordination. O,Reilly, (2014).
5. Apache Kafka project: A distributed streaming platform. «http://kafka.apache.org/» (2017)
6. Apache Spark: A fast and general engine for large-scale data processing. «http://spark.apache.org/»(2017).
7. MEMSQL:Tutorials overview «https://docs.memsql.com/tutorials/v5.8/tutorials-overview/» (2018).
8. Huppler K.: The Art of Building a Good Benchmark. In: Nambiar R., Poess M. (eds) Performance Evaluation and Benchmarking. TPCTC 2009. Lecture Notes in Computer Science, vol 5895. Springer, Berlin, Heidelberg.
9. Nambiar R., Poess M. (2018) Industry Standards for the Analytics Era: TPC Roadmap. Lecture Notes in Computer Science, vol 10661. Springer, Cham (2018).
10. Li, M., Tan, J., Wang, Y. et al. SparkBench: a spark benchmarking suite characterizing large-scale in-memory data analytics. Cluster Comput (2017).
11. B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with YCSB, in Proceedings of the 1st ACM symposium on Cloud computing, ser. SoCC '10, 2010, pp. 143–154.
12. Curino, C., Difallah, D.E., Pavlo, A., Cudré-Mauroux, P. (2012). Benchmarking OLTP/Web Databases in the Cloud: the OLTP-Bench Framework. 4th International Workshop on CloudData Management (CloudDB 2012), Maui, HI, USA. 17-20.
13. T. G. Armstrong, V. Ponnekanti, D. Borthakur, and M. Callaghan. Linkbench: a database benchmark based on the facebook social graph. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. ACM, 2013, pp. 1185–1196.
14. A. Ghazal, T. Rabl, et al. 2013. BigBench: towards an industry standard benchmark for big data analytics. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. ACM, New York, NY, USA, 1197-1208.
15. Baru C. et al. Discussion of BigBench: A Proposed Industry Standard Performance Benchmark for Big Data. Lecture Notes in Computer Science, vol. 8904. Springer (2015).
16. Wanling Gao et al (2018) Bigdatabench: A Dwarf-Based Big Data And Ai Benchmark Suite. Technical Report No. ACS/Ssl-2018-1. January 27, 2018.
17. Bhandarkar M. AdBench: A Complete Benchmark for Modern Data Pipelines. Lecture Notes in Computer Science, vol 10080. Springer, (2017).
18. Shukla A., Simmhan Y. (2017) Benchmarking Distributed Stream Processing Platforms for IoT Applications. Lecture Notes in Computer Science, vol 10080. Springer.
19. ORACLE: "Overview of the Event Processing Language (EPL)" «https://docs.oracle.com/cd/E13213_01/wlevs/docs30/epl_guide/overview.html» (2008).