Ejecución y monitoreo de procesos de negocios distribuidos entre diferentes motores de Bonita OS

Leonardo Damián Karabogosian[2], Patricia Bazán[1], Jose Martinez Garro[2]
1. LINTI – Facultad de Informática – Universidad Nacional de La Plata
2. Facultad de Informática - Universidad Nacional de La Plata
leonardo.karabogosian@gmail.com, pbaz@info.unlp.edu.ar, josemartinezgarro@yahoo.com.ar

Resumen

La combinación de las tecnologías asociadas a *Cloud Computing* y BPM (*Business Process Management*) modifica aspectos tanto de diseño como de ejecución de los procesos de negocios. Los ambientes distribuidos en el contexto de los procesos favorecen el rendimiento y proponen incorporar el concepto de descomposición de procesos, permitiendo que los mismos se ejecuten tanto en un entorno *cloud* o embebido.

Si bien la descomposición de procesos es un tema abordado en los últimos años, el monitoreo de dichos procesos no ha sido demasiado explorado aún.

Este trabajo propone una implementación de una arquitectura para un sistema de monitoreo de procesos distribuidos utilizando *Bonita Open Solution* como motor de procesos, su API y el uso de conectores personalizados.

Palabras clave: *Monitoreo, Cloud Computing, BPM, Bonita Open Solution, Web Services, REST*

Contexto

El presente es un trabajo de fin de carrera de Licenciatura en Sistemas de la Facultad de Informática de la UNLP, del alumno Leonardo Karabogosian, dirigido por la Mg. Patricia Bazán y con el asesoramiento profesional del Lic. José Martinez Garro.

Introducción

BPM basado en *Cloud Computing* (denominado *Business Process as a Service* - BPaaS), al igual que varios modelos de servicios en *cloud*, permite a los usuarios utilizar un BPMS situado en la "nube" bajo una modalidad de "pago por uso", en lugar de enfrentarse a grandes costos de inversión en software, hardware y mantenimiento. Podríamos considerar que BPaaS es un modelo de servicio en el *cloud* donde las aplicaciones que se ofrecen son del tipo procesos de negocio o *workflows*.

Este modelo de servicio necesita de dos componentes principales para el desarrollo y ejecución de sus procesos, que son utilizados para lograr las funcionalidades de un *workflow* tal como se ejecutaría en un sistema embebido:

- BPMS (*Business Process Management System*): herramientas para diseñar, ejecutar, monitorear y optimizar los procesos de negocios.
- BAM (*Business Activity Monitoring*): herramienta para monitorización de ejecución de procesos de negocio en tiempo real.

Utilizar un BPMS en el *cloud* tiene la desventaja de que los usuarios pierden el control de los datos sensibles que hacen al funcionamiento de las organizaciones. Además, el poder de cómputo del *cloud* no estaría aprovechado cuando las actividades de los procesos que se despliegan en ese entorno no requieren de alta capacidad en su performance, pero sí tienen un frecuente acceso a datos, lo cual intensifica el tráfico [1].

Para conciliar la privacidad de datos en un sistema embebido con el poder de cómputo de un sistema de *cloud*, se ha investigado en [2] un modelo de distribución, denominado PAD, en el que el arquitecto de procesos de negocio puede separar un proceso de acuerdo a los beneficios que brindan cada una de estas ubicaciones.

En el modelo PAD (Proceso-Actividad-Datos) el motor de procesos, las actividades involucradas en el mismo y sus datos están distribuidos. Dicho modelo define cuatro patrones de distribución, que podemos observar de forma completa en [2] y [3], y sintetizarlo en:

1. Sistema BPM completamente embebido.
2. BPM embebido con componentes en el cloud.
3. BPM de Cloud con componentes embebidos.
4. BPM basado en Cloud.

A partir del modelo PAD, proponemos un quinto patrón en el cuál el motor de procesos, las actividades y los datos se despliegan en ambos sitios, es decir, tanto en el usuario final como en la nube. Esta solución presenta el beneficio de que, al ubicar un motor de procesos en el *cloud* y otro en el sistema embebido, permite regular el flujo de control y el flujo de datos de las actividades en ambos sitios. Pero para poder ejecutar un proceso de negocio en dos motores de proceso separados, el mismo debe ser dividido en dos procesos individuales: uno a ejecutarse en el motor de procesos embebido y otro en el motor situado en el *cloud*. [3]

El monitoreo del proceso de negocio es más complicado ahora, debido a que el mismo ha sido subdividido en dos partes. Como solución a presentar, se desarrolla una herramienta de monitoreo para el proceso original, mediante la combinación de los detalles de monitoreo de los procesos individuales y basada en la arquitectura planteada en [1].

a. Bonita OS y su API REST

La implementación de la arquitectura para el monitoreo de procesos distribuidos mencionada se realizó utilizando Bonita OS debido a que es un BPMS de código abierto, en el cual podemos modelar nuestros procesos de negocio, exportarlos y ejecutarlos en un ambiente de producción, y posee un portal en donde el usuario puede interactuar con aquellas actividades que le son asignadas.

Bonita es de fácil instalación, y se ejecuta dentro de un servidor de aplicaciones (*Tomcat* o *JBoss*) que se lanza con mínimas configuraciones. Bonita OS se compone de 4 módulos:

-Bonita Execution Engine (BEE): es el motor BPM de Bonita, y se encarga de la ejecución y despliegue de los procesos.

-Bonita Studio: es la aplicación utilizada para diseñar los procesos (según la notación BPMN) y los conectores.

-Bonita Form Builder: es la aplicación encargada de mostrar los formularios para el ingreso de información dentro de las actividades.

-Bonita User Experience (XP): es la aplicación encargada de la gestión de los procesos de negocio desplegados.

Para ejecutar los procesos de Bonita desde una aplicación externa, es necesario tener acceso al BEE a través de su API REST. Esta API se compone de una serie de clases Java [5] para controlar la aplicación completa tanto de manera local como remota [4], y a su vez para recuperar la información de aquellos procesos que se encuentran desplegados, con el fin de unir la información requerida para el monitoreo.

Además del acceso a las instancias remotas, es necesario almacenar la información correspondiente a cada una de las instancias desplegadas tanto en el servidor local como en el remoto. Para unir ambas funcionalidades se desarrolló un conector que pueda ejecutar un proceso en otro servidor, y a su vez, almacenar la información de todas las instancias creadas.

b. Creación de un conector y persistencia en la base de datos

Los conectores en Bonita se crean utilizando la aplicación Bonita Studio mediante un asistente interno. Un conector está dividido en dos partes: la definición, que controla las interfaces externas del mismo (visibles al usuario y al BEE); y la implementación, que consiste en un archivo XML (describiendo identificadores, versiones y dependencias del mismo), así como una clase Java, en la cual el creador del conector escribe el código de su comportamiento en una interfaz de desarrollo similar a Eclipse [6].

La tarea de nuestro conector consiste en comunicarse con otro motor de ejecución de Bonita situado en un servidor remoto, crear una instancia de proceso en ese servidor, enviar aquellos datos necesarios como entrada a la nueva instancia, y guardar los identificadores de las instancias locales y remotas en una base de datos para luego poder relacionar ambas instancias al momento de realizar el monitoreo.

Para instanciar un proceso situado en otro servidor, es necesario que el conector, localizado en el servidor local, se ejecute en el servidor remoto. Esto es posible realizando un cambio de contexto para que el conector conozca la dirección IP del servidor remoto, así como también sus parámetros de acceso (usuario y contraseña). Este cambio de contexto y la autenticación al servidor se realizan a través de configuraciones JAAS [7]. Previo a esto se debe suministrar al conector el nombre del proceso remoto y su versión, para poder así instanciar el proceso y continuar con la ejecución. Una vez creada la instancia en el servidor remoto, los datos correspondientes tanto a la instancia local como remota, se almacenan en una base de datos local que se utilizará para realizar el seguimiento de todo el proceso. En nuestra solución se plantea el uso de una base de datos MySQL, por lo que se requerirá importar el *driver* de Java (com.mysql.jdbc.Driver) para interactuar con la misma.

El conector desarrollado debe restablecer el contexto en que se encontraba el sistema previo a su ejecución.

c. Arquitectura del sistema de monitoreo y la aplicación Web

La arquitectura del sistema de monitoreo puede verse en la Figura 1, en la cual podemos observar el flujo de la información entre los distintos componentes del sistema. Observando esta Figura, en los nodos 1 y 2 el conector hace uso del *driver* de MySQL para almacenar la información de las instancias, y de la API (Java) de Bonita para crear la instancia del proceso en el servidor remoto; en 3 la aplicación utiliza servicios web localizados en otra aplicación remota para obtener la información correspondiente a las actividades de las instancias. En 4, dicha aplicación remota hace uso de la API REST de Bonita para poder responder a la solicitud.

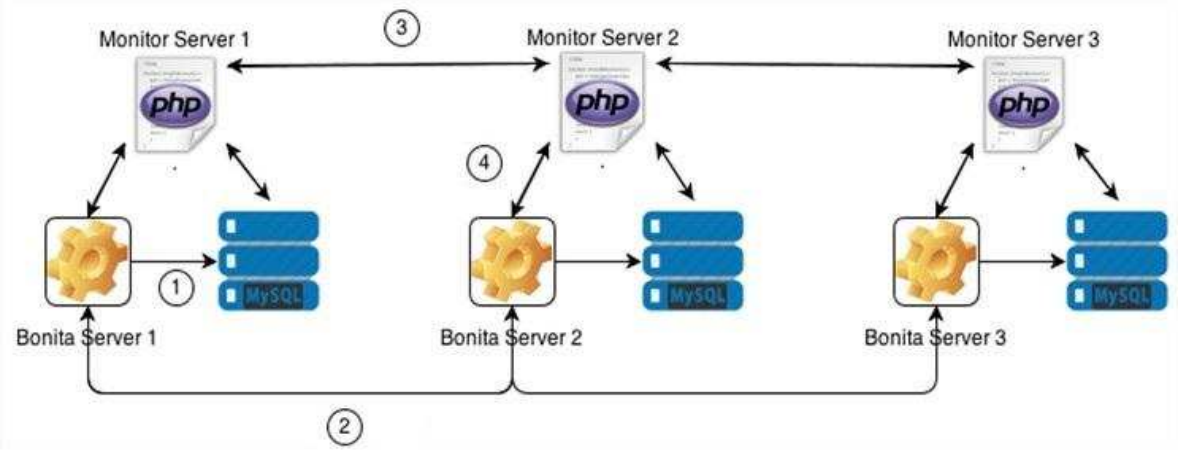

Figura 1: Arquitectura del sistema de monitoreo

Si bien son visibles solo 3 servidores, la arquitectura es extensible. Sin embargo, en caso de existir un encadenamiento de procesos entre los servidores 1, 2 y 3, el servidor 3 es totalmente invisible al servidor 1, siendo el 2 su intermediario.

Para recopilar la información remota a través de servicios web y graficar el monitoreo completo, se desarrolló una aplicación Web utilizando las tecnologías PHP, HTML y CSS, junto con las siguientes librerías complementarias:

- JQuery y JTable: con fines visuales y de diseño.
- NuSOAP: para la comunicación entre las aplicaciones de monitoreo y el motor de Bonita. [8]
- SimpleXML: como estándar para el formato de la información de las instancias. [9]
- GraphViz: para diagramar el proceso de negocio distribuido unificado.

La solución desarrollada permite monitorear cada una de las instancias que han sido creadas utilizando el conector, mostrando además de la instancia local y remota, cada una de las actividades involucradas en las mismas. Se obtiene, entre otros datos el detalle del estado, inicio, finalización y actor de las actividades.

Para obtener una visión unificada del monitoreo distribuido se genera un archivo XML que consolida cada uno de los datos obtenidos de las instancias remotas. Dicha tarea se realiza identificando los tipos de actividades que se encuentran en las instancias, es decir, si son *tokens* de inicio o de fin, tareas, compuertas, flujos, si tienen conectores, actores asociados, entre otros, y una vez conocida la información para generar el proceso original, se utiliza la librería GraphViz [10], la cual nos permite graficar las actividades como nodos y el flujo de control como aristas direccionales, logrando una visualización del proceso de negocio original similar a la que puede realizarse con la aplicación de Bonita Studio.

Líneas de investigación, desarrollo e innovación

El presente trabajo continúa una línea de investigación iniciada durante el año 2013 en torno a abordar aspectos que resignifiquen los conceptos de gestión de procesos de negocio a la luz de las nuevas tendencias tecnológicas. La alta disponibilidad y la creciente conectividad están llevando a las organizaciones a

optar por un modelo de *Cloud Computing*.

Al ser BPM un paradigma naturalmente integrador, existen conceptos como dinamismo de *workflows* y servicios móviles y dinámicos que se replantean al entrar en un contexto de *cloud*.

Resultados y Objetivos

El resultado de este trabajo es la implementación de un sistema de monitoreo en un ambiente híbrido, utilizando motores de procesos de Bonita Open Solution. El mismo es un BPMS que dispone de una API basada en una arquitectura REST, y a la vez es software libre y *open source*.

Este desarrollo permitió obtener una implementación concreta de un aspecto de la distribución de procesos como es el monitoreo. Esta implementación se integra con otros trabajos en curso que extienden otros aspectos vinculados a la gestión por procesos, como la minería de procesos distribuidos.

Formación de Recursos Humanos

BPM ha cobrado importancia dentro del área de tecnología informática, la cual en los últimos años ha evolucionado desde el concepto de producto hacia el paradigma de servicios y soluciones. El presente artículo ilustra una línea de investigación iniciada en el año 2008, donde no solo se cubre el desarrollo de procesos BPM sino también la mejora continua de los mismos y la ampliación de sus ambientes de ejecución. En dicha línea se están formando alumnos para desarrollar su tesina e interactuar con docentes e investigadores formados, incorporando BPM y sus herramientas de soporte como línea de acción para solución de problemas reales.

Referencias

[1] Evert F. Duipmans. "Business Process Management in the cloud with data and activity distribution". Faculty of electrical engineering, mathematics and computer science software engineering. University of Twente. EWI/SE - 2012-002. November 2012

[2] Han YB, Sun JY, Wang GL et al. A cloud-based BPM architecture with user-end distribution of non-compute-intensive activities and sensitive data. Journal of computer science and technology. Nov. 2010. DOI 10.1007/s11390-010-1092-5

[3] José Martinez Garro, Patricia Bazan. "Monitoreo de procesos distribuidos en el cloud. Una propuesta arquitectónica" WBPM 2013: Chilean Workshop on Business Process Management (BPM). Noviembre, 2013. Temuco, Chile. Noviembre 2013. ISBN 978-956-7019-95-

[4] Documentación BonitaSoft (Julio 2013) - http://documentation.bonitasoft.com/

[5] API REST de BonitaSoft (Agosto 2013) - http://documentation.bonitasoft.com/javadoc/rest/5.10/API/index.html

[6] "Creación de un conector en Bonitasoft" – Tutorial (Agosto 2013) http://documentation.bonitasoft.com/creating-connector

[7] Java Authentication and Authorization Service (JAAS) (Agosto 2013) - http://docs.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html

[8] NuSOAP (Agosto 2013) - http://nusoap.sourceforge.net/

[9] SimpleXML (Agosto 2013) - http://php.net/simplexml

[10] GraphViz (Septiembre 2013) - http://www.graphviz.org/