Hive: Framework de Sincronización de Objetos en la Nube para Sistemas Distribuidos Multiplataforma

Matías Teragni, Gonzalo Zabala, Ricardo Morán, Sebastián Blanco.
Centro de Altos Estudios en Tecnología Informática
Facultad de Tecnología Informática
Universidad Abierta Interamericana
Av. Montes de Oca 745, Ciudad Autónoma de Buenos Aires, República Argentina
(+54 11) 4301-5323; 4301-5240; 4301-5248
{ Matias.Teragni, Gonzalo.Zabala, Ricardo.Moran, Sebastian.Blanco}@uai.edu.ar

Resumen

El propósito de este proyecto es diseñar e implementar un conjunto de librerías que permitan compartir su estado de ejecución a distintos programas que se encuentran corriendo en distintas maquinas. Esta funcionalidad será extendida a diversos lenguajes ampliamente utilizados, permitiendo la interoperabilidad entre distintos tipos de aplicaciones (web, desktop, mobile) y diversas plataformas.

Palabras clave: Programación Distribuida, Internet of things, Cloud robotics, multiplataforma

Contexto

El presente proyecto será radicado en el Centro de Altos Estudios en Tecnología Informática, dependiente de la Facultad de Tecnología Informática de la Universidad Abierta Interamericana. El mismo se encuentra inserto en la línea de investigación “Nuevas Tecnologías para Internet”. El financiamiento está compartido entre el CONICET y la misma Universidad por partes iguales.

Introducción

Se entiende a un sistema distribuido como un conjunto de computadoras independientes, cada una con su propia memoria y capacidad de procesamiento, que se encuentran interconectadas por medio de una red, y sobre las cuales opera un conjunto de software que colabora para lograr un comportamiento complejo como un todo.

Los sistemas distribuidos pueden proveer, dada su naturaleza, un conjunto de beneficios, desde resistencia a fallos mediante replicación, hasta la capacidad de ejecutar procesos sumamente complejos en computadoras económicas mediante la partición del problema en partes más simples.

Debido a estos beneficios, y que algunos sistemas requieren por su dominio del problema estar distribuidos, se volvieron populares un conjunto de tecnologías con el objetivo de facilitar la construcción de este tipo de sistemas interconectados, admitiendo abiertamente la llamada a procedimientos remotos (corba, web services, http, etc). A los sistemas que admiten esta diferencia se los considera fuertemente acoplados. Esto implica que a la hora de crear, o programar estos sistemas, no solo se requiere trabajo adicional, sino que también aumenta la complejidad del sistema, dificultando los procesos de testing, debugging, y mantenimiento del software.

En paralelo a estos desarrollos se idearon formas de interconectar sistemas de forma transparente, es decir, que comparten variables, memoria, o estado, sin la necesidad de que el programador cambie como se utilizan las mismas en su ámbito particular. Estos métodos de distribución se conocen como bajamente acoplados, y presentan como principal beneficio que no genera trabajo adicional para los desarrolladores.

Estos últimos no fueron exitosos por dos grandes motivos. En principio la velocidad de las redes de telecomunicación era insuficiente para reflejar la velocidad y extensión de los cambios en la memoria de un programa. Y para empeorar el problema, la mayoría de las implementaciones se basan en la definición de lenguajes de programación propietarios, que definen reglas de interacción arbitrarias. Es decir, se requiere desarrollar nuevas versiones del sistema, contratando a programadores expertos en estos lenguajes, para lograr que un software ya desarrollado se pueda utilizar en un ambiente distribuido.

El presente trabajo tiene como objetivo desarrollar un conjunto de librerías que permitan mantener las ventajas de las arquitecturas distribuidas sin tener que utilizar semánticas especiales y complejas para su especificación.

Líneas de Investigación, Desarrollo e Innovación

Este proyecto en particular tiene dos fuentes de complejidad. En primer lugar el objetivo de proveer la capacidad de distribuir un ambiente de objetos, y en segundo el que esto se pueda realizar en múltiples plataformas, lenguajes y de forma transparente para el programador.

Dado que las posibles ramificaciones de esta tecnología son múltiples, plantea un proceso iterativo e incremental. Estando cada iteración compuesta por las siguientes actividades:

- Identificación de problemas no resueltos: Se definirán un conjunto de escenarios o funcionalidades que se desee explorar, o errores encontrados en iteraciones anteriores, generando un listado que indicará los objetivos que se desean cumplir con la iteración

- Planteo teórico sobre una solución: Se evaluará para el listado de problemas no resueltos cuáles se abordarán en esta iteración, y se diagramará una solución teórica para estas dificultades.

- Definición de casos de prueba: Se definirán formas de probar las funcionalidades a ser abordadas durante esta iteración.

- Implementación de un prototipo incremental: Se implementará el código que lleve a cabo las soluciones teóricas planteadas, realizando pruebas unitarias sobre el mismo y de haberse realizado iteraciones previas se incorporará a aquello ya desarrollado.

- Evaluación de avances alcanzados: Se tomarán los datos recaudados de las pruebas, para concluir hasta qué punto se lograron resolver los objetivos planteados para esta iteración. Esto permitirá definir cuál es el estado actual de la solución, y planteará la base para la próxima iteración.

Los hitos a cumplir en pos de realizar dicha solución son:

- Crear un sistema que permita compartir estado entre aplicaciones en un solo lenguaje de programación

- Extender la funcionalidad incorporando otros lenguajes y plataformas de uso masivo

- Demostrar el uso y las ventajas del proyecto mediante la implementación de casos de ejemplo basados en sistemas reales

- Incorporar la sincronización de comportamiento a lo ya existente.

- Demostrar las ventajas a la hora de construir sistemas complejos mediante la colaboración provista por esta herramienta mediante el uso de ejemplos basados en sistemas reales

Resultados y objetivos

Diseñar e implementar un esquema de persistencia cloud que permita mantener el estado de una aplicación orientada a objetos

Construir las librerías y APIs que permitan sincronizar transparentemente el estado en aplicaciones en javascript

Extender la funcionalidad a lenguajes ampliamente utilizados (C#, Java), permitiendo la interoperabilidad entre distintos tipos de aplicaciones (web, desktop) y diferentes plataformas.

Formación de Recursos Humanos

El equipo de trabajo está conformado por un investigador adjunto del Centro de Altos Estudios en Tecnología Informática (CAETI) quien ejerce el rol de director del proyecto, dos doctorandos, y un ayudante alumno de la Facultad de Tecnología Informática de la Universidad Abierta Interamericana.

Referencias

- Coulouris, George; Jean Dollimore; Tim Kindberg; Gordon Blair (2011). Distributed Systems: Concepts and Design (5th Edition). Boston: Addison-Wesley.
- LeLann, G. (1977). "Distributed systems - toward a formal approach,". Information Processing. 77: 155·160. – via Elsevier.
- Andrews, Gregory R. (2000), Foundations of Multithreaded, Parallel, and Distributed Programming, Addison–Wesley, ISBN 0-201-35752-6.
- Christian Cachin; Rachid Guerraoui; Luís Rodrigues (2011), Introduction to Reliable and Secure Distributed Programming (2. ed.), Springer, ISBN 978-3-642-15259-7