Depuración y Evolución de Aplicaciones Distribuidas y Concurrentes

V Jornada de Socialización de la Investigación en Icesi 2012

29 y 30 de agosto de 2012

David Durán
Universidad Icesi
Grupo de Investigación I2T
dduran@icesi.edu.co

Jhonny Ocampo
Universidad Icesi
Grupo de Investigación I2T
jocampo@icesi.edu.co

Hugo Arboleda
Universidad Icesi
Grupo de Investigación I2T
hfarboleda@icesi.edu.co

Este trabajo presenta una propuesta de un lenguaje que será la base para un *framework* de eventos que soporta la detección de patrones complejos en sistemas distribuidos, utilizando autómatas para modelar los patrones complejos de interacción entre los nodos que participan en el sistema distribuido. Por medio de la presentación de diferentes errores concurrentes comunes que ocurren o han ocurrido en aplicaciones industriales de esta índole, como el *deadlock* o los *dataraces*, se proponen soluciones a este tipo de inconvenientes utilizando el lenguaje propuesto, demostrando su utilidad y aplicabilidad. En concreto, se presentan las siguientes contribuciones: i) el diseño de un lenguaje de programación orientado a eventos con soporte para declaración, ejecución, detección y coordinación de patrones de eventos complejos en sistemas distribuidos, ii) propuestas de definición de autómatas utilizando el lenguaje propuesto para detectar los errores concurrentes comunes identificados en aplicaciones distribuidas industriales y iii) la implementación de un *kernel* para soportar las abstracciones del lenguaje por medio de una extensión a la librería KETAL (Benavides Navarro, Barrera, Garces, & Arboleda, 2011), la cual define mecanismos de sincronización de eventos.

Introducción

Un sistema distribuido consiste en una colección de computadores autónomos llamados nodos o hosts que están conectados a través de una red y un middleware de distribución, los cuales permiten a los nodos coordinar sus actividades y compartir los recursos del sistema, de modo que los usuarios perciban el sistema como uno solo (Tanenbaum A. S., 1996) y a (Tanenbaum & Steen, 2006). Ejemplos de estos sistemas son los llamados sistemas intensivos de software, los cuales han tenido un impacto bastante grande en la sociedad en los últimos años.

Objetivo

Actualmente, las interacciones de múltiples componentes distribuidos y la presencia de múltiples procesos concurrentes hacen la programación de aplicaciones distribuidas una tarea compleja. El objetivo de este trabajo fue diseñar un lenguaje de programación orientado a eventos con soporte para declaración, ejecución, detección y coordinación de patrones de eventos complejos en sistemas distribuidos utilizando autómatas, ordenamiento de eventos basados en la causalidad y la composición síncrona y asíncrona de eventos.

Lenguaje de Eventos para Aplicaciones Distribuidas

El lenguaje propuesto soporta programación orientada a aspectos en un contexto distribuido donde un *pointcut* puede monitorear eventos en diferentes hosts.

$$
\begin{array}{lll}
Ep & ::= & \text{call}(ESig) \\
 & | & \text{host}(Group) \mid \text{on}(Group[, Select]) \\
 & | & \text{args}(\{Arg\}) \\
 & | & \text{eq}(JExp, JExp) \mid \text{if}(JExp) \\
 & | & Ep \mid\mid Ep \mid Ep \;\&\&\; Ep \mid !Ep \\
Group & ::= & \{ Hosts \} \\
Hosts & ::= & \text{localhost} \mid \text{jphost} \mid \text{"}Ip{:}Port\text{"} \\
 & | & GroupId \\
GroupId & ::= & String \\
Select & ::= & JClass
\end{array}
$$

Los *poincuts* también pueden ser combinados usando operaciones lógicas como la negación, conjunción y disyunción. Para los eventos complejos se utiliza un autómata modelándolo con una sintaxis bien definida.

$$
\begin{array}{lll}
Seq & ::= & [Id{:}]\ \text{seq}(\{Step\}) \mid \text{step}(Id,Id) \\
Step & ::= & [Id{:}]\ Ep\ [>Target\ ] \\
Target & ::= & Id \mid Id \mid\mid Target
\end{array}
$$

La administración de reacciones también es contemplada como parte del lenguaje.

$$
\begin{array}{lll}
Rc & ::= & [\text{asyncex}]\ \text{reaction}(\{Par\}) : EcAppl\ \text{'\{'}\ \{Body\}\ \text{'\}'} \\
EcAppl & ::= & Id(\{Par\}) \\
Body & ::= & JStmt \\
 & | & \text{addGroup(Group)} \mid \text{removeGroup(Group)}
\end{array}
$$

Conclusiones

El lenguaje de programación orientado a eventos que hemos desarrollado permite la declaración, ejecución, detección y coordinación de patrones de eventos complejos en sistemas distribuidos. Esta permite el uso de autómatas finitos deterministas con guardas para la captura de eventos complejos, y utiliza además, los relojes vectoriales de Mattern para el ordenamiento causal de estos eventos. Mediante la extensión a la librería KETAL, en la cual se implementó una nueva característica para el llamado síncrono y asíncrono con futuros para el envió de eventos, se mejoró el soporte para la detección de eventos compuestos. La utilización de autómatas deterministas para modelar este tipo de problemas concurrentes en ambientes distribuidos resulta conveniente porque, además de que permite concebir y crear de manera sencilla la interacción entre eventos, la complejidad del código resultante depende directamente del orden de ocurrencia único de los eventos para que se presente el problema abordado, independientemente de si dichos eventos hacen parte de diferentes procesos concurrentes o no.

Referencias

Benavides Navarro, D. L., Barrera, A., Garces, K. O., & Arboleda, H. (2011). Detecting and Coordinating Complex Patterns of Distributed Events with KETAL. *Proc. of the 2011 Latin American Conference in Informatics (CLEI) , 281*, 127-141.

Tanenbaum, A. S. (1996). *Sistemas Operativos Distribuidos* (Primera ed.). Prentice Hall.

Tanenbaum, A., & Steen, M. V. (2006). *Distributed Systems: Principles and Paradigms* (2nd Edition ed.). Upper Saddle River, NJ, USA: Prentice Hall, Inc.