Propuesta de investigación: Marco Teórico Inicial y Objetivos.

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Maestría en Computación
MC-7205 Tema Selecto de Investigación

CARLOS MARTÍN FLORES GONZÁLEZ, Carné: 2015183528

Índice

1. PERFORMANCE EVALUATION OF COMPONENT-BASED SOFTWARE SYSTEMS: A SURVEY[1]

Durante los últimos diez años, los investigadores han propuesto muchos enfoques para evaluar el rendimiento (tiempos de respuesta, *throughput*, utlización de recursos) de sistemas de software basados en componentes. Estos enfoques lidian con predicción de rendimiento y medición del rendimiento. Los primeros analizan el rendimiento esperado de un diseño de software basado en componentes para evitar problemas de rendimiento en la implementación del sistema, lo que podría llevar a costos substanciales para rediseñas la arquitectura. Los otrosanalizan el rendimiento observable de sistemas de software basados en componentes implementados para entender sus propiedades, determinar su capacidad máxima, identificar componentes críticos y para remover cuellos de botella.

Métodos de evaluación de rendimiento. Se agruparon los enfoques en dos grandes grupos: enfoques principales que proporcionan procesos de evaluación de rendimiento completo y enfoques suplementarios que se centran en aspectos específicos como medición de componentes individuales o modelaje de las propiedades de rendimiento de los conectores de un componente.

Enfoques principales.

- Enfoques de predicción basados en UML:
  - CB-SPE - *The Component-Based Software Performance Engineering*
- Enfoques de predicción basados en meta-modelos propietarios
  - CBML - *The Component-Based Modelling Language*
  - PECT - *The Prediction Enabled Component Technology*
  - COMQUAD - *Components with Quantitative properties and Adaptivity*
  - KLAPPER
  - ROBOCOP
  - PALLADIO
- Enfoques de predicción centrados en *middleware*
  - NICTA
- Enfoques basados en especificaciones formales
  - RESOLVE
  - HAMLET
- Enfoques de monitoreo para sistemas implementados
  - COMPAS
  - TESTEJB
  - AQUA
  - PAD

Enfoques Suplementarios.

- Enfoques de monitoreo para componentes implementados
  - RelCAM
  - COMAERA
  - BYCOUNTER
- Enfoques de predicción para conectores de componentes
  - Verdickt
  - Grassi
  - Becker
  - Happe

2. THE PALLADIO COMPONENT MODEL FOR MODEL-DRIVEN PERFORMANCE PREDICTION[2]

El el desarrollo de software basado en componentes (CBSE, por sus siglas en inglés) la idea central es construir sistemas de software complejo uniendo componentes básicos. El objetivo inical de CBSE fue incremetar el nivel de reutilización. Sin embargo, estructuras compuestas también pueden aumentar el nivel de predictibilidad del sistema durante etapas tempranas de desarrollo, esto porque modelos certificados de componentes individuales puee ser compuestos, permitiendo a los arquitectos de software razonar sobre la estructura compuesta. Esto es importante para las propiedades funcionales, pero también para las propiedades extra funcionales como rendimiento y confiabilidad.

Los métodos para predicción de rendimiento y confiabilidad de sistemas de software en general son limitados y raramente usados en la industria. Este reto es aún mayor en CBSE debido a que varios roles independientes están involucrados en la creación del sistema. Muchos métodos existentes para predicción de CBSE requieren que los arquitectos de software modelen el sistema basados en especificaciones de un solo componente. Luego se asume que el arquitecto proporcionará información faltante. Otros enfoques dejan de lado factores que afectan el rendimiento percibido de un componente de software tal y como la influencia de servicios externos, cambio de recursos de su ambiente o diferentes parámetros de entrada. Sin embargo, para predicciones precisas, todas estas dependencias tiene que hacerse explícitas en la especificación del componente.

Con *Palladio*[1] *Component Model* (PCM) un meta-modelo provee la especificación de información relevante del rendimiento de una arquitectura basada en componentes. El diseño y análisis de estos meta-modelos son los primeros que explícitamente incluyen todos los factores que influencian el rendimiento de un componente de software como lo son: la implementación, rendimiento de servicios externos, rendimiento del ambiente de ejecución y el perfil de uso. El modelo es diseñado con especial énfasis en la predicción de atributos de calidad de servicio como el rendimiento y la confiabilidad.

En el proceso de CBSE(figura 1) los autores distinguen cuatro tipos de roles involucrados en la producción de artefactos de un sistema de software: *desarrolladores de componentes* especifican e implementan los componentes, *arquitectos de software* ensamblan componentes para construir aplicaciones, *system deployers* modela los recursos del ambiente y luego su asignación, *expertos del dominio de negocio* que están familiarizados con los usuarios del sistema y proporcionan modelos de su uso. El modelo completo del sistema puede ser derivado a partir de modelos parciales especificados por cada uno de los roles para que luego las propiedades extra funcionales puedan ser predecidas.

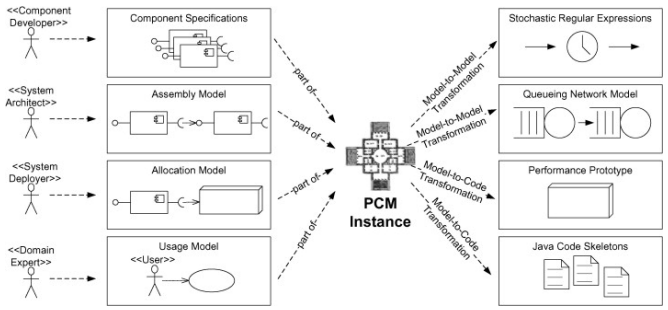

Fig. 1. Proceso CBSE propuesto por PCM

[1]Toma su nombre del arquitecto renacentista Andrea Palladio (1508 - 1580) quien de cierta forma, intentó predecir el impacto estético de sus construcciones por adelantado.

3. PERFORMANCE-ORIENTED DEVOPS: A RESEARCH AGENDA[3]

DevOps es una tendencia hacia una estrecha integración entre equipos desarrollo y operaciones. La necesidad de tal integración es orientada por el requerimiento de adaptar aplicacione empresariales continuamente a los cambios del ambiente del negocio. El rendimiento describe las propiedades del sistema con respecto a su ejecución y uso de recursos. Métricas comunes son tiempo de respuesta, *throughput* y la utilización de los recursos. Los objetivos de rendimiento para aplicaciones empresariales son típicamente definidos al establecer cotas superiores/inferiores para estas métricas y transacciones de negocio específicas.

3.1. Actividades de administración del rendimiento

3.1.1. Evaluación del rendimiento basado en modelos.

- La representación de la memoria principal y del recolector de basura aún no están explícitamente integrados ni considerados en los modelos de rendimiento
- La selección de técnicas apropiadas de solución requiere de mucha experiencia

3.1.2. Extracción de modelos de rendimiento y de cargas de trabajo.

- La precisión de los modelos puede llegar a expirar si no son actualizados cuando hay cambios. Mecanismos de detección son requeridos para aprender cuando los modelos son antiguos y hay que actualizarlos.
- La extracción de capacidades de rendimiento se basa en una combinación de software y los recursos de hardware en los que se implementa. Este enfoque combinado es compatible con la precisión de la predicción, pero está menos calificado con respecto a la portabilidad de los conocimientos a otras plataformas. Una dirección de investigación futura podría ser extraer modelos separados (por ejemplo, *middleware* y modelos de aplicación separados).

3.2. Ingeniería de rendimiento de software durante el desarrollo

3.2.1. Modelos de rendimiento durante etapas diseño .

- Los retos de utilizar modelos de rendimiento en etapas tempranas de desarrollo es que usualmente es difícil validar la precisión de los modelos hasta que un sistema en ejecución exista. Las predicciones de rendimiento basadas en suposiciones, entrevistas y pruebas previas pueden ser también imprecisas y por tanto las decisiones que se hagan a partir de estas predicciones.

Sobre este reporte.

4. PERFORMANCE ENGINEERING FOR MICROSERVICES: RESEARCH CHALLENGES AND DIRECTIONS[4]

Los microservicios complementan enfoques como DevOps y entrega continua(CD por sus siglas en inglés) en relación en términos de arquitectura de software. Junto con este estilo de arquitectura, otras tecnologías importantes para *deployment* como, virtualización basada en contenedores y soluciones de orquestación de contenedores han emergido. Estas tecnologías permiten explotar plataformas en la nube, proporcionando altos niveles de escalabilidad, disponibilidad y portabilidad para microservicios. A pesar del hecho de que es una necesidad inherente contar con escalabilidad y elasticidad, la ingeniería de rendimiento para microservicios hasta ahora ha tenido poca atención por parte de las comunidades tanto de microservicios como de comunidades de investigación de ingeniería de rendimiento. Un gran cuepro de conocimiento y buenas prácticas para ingeniería de rendimiento para desarrollo tradicional de software y arquitecturas está disponible. Sin embargo, sus aplicaciones en DevOps imponen tanto retos como oportunidades.

4.1. Retos de investigación

4.1.1. Pruebas de rendimiento.

- Reemplazar y compensar pruebas extensivas de integración y de sistema por un control detallado de los entornos de producción
- Alinear las pruebas de rendimiento y pruebas de regresión de rendimiento con prácticas de CD, por ejemplo, acelerar las estas de pruebas correspondientes.
- Selección dinámica y semi automática de pruebas de rendimiento

4.1.2. Monitoreo.

- Instrumentación para monitoreo distribuido arquitecturas microservicios políglotas
- Métricas adicionales para monitorear microservicios
- Técnicas de detección precisa de anomalías en arquitecturas de microservicios

4.1.3. Modelado del rendimiento. De momento no existen enfoques para modelado del rendimiento de microservicios

- Adoptar modelos de rendimiento para casos de uso como planeamiento de capacidad, confiabilidad y resiliencia
- Buscar abstracciones de modelado apropiadas
- Extracción automática de modelos de rendimiento
- Aprender del comportamiento de la infraestructura e integrarlo en los modelos de rendimiento

5. AN INDUSTRIAL CASE STUDY OF PERFORMANCE AND COST DESIGN SPACE EXPLORATION[5]

Determinar la compensación(*trade-off*) entre rendimiento y costo de un sistema de software distribuido es importante ya que permite cumplir los requerimientos de rendimiento de una forma rentable. La gran cantidad de alternativas de diseño para tales sistemas usualmente llevan a los arquitectos de software a seleccionar soluciones subóptimas, las cuales puede desperdiciar recursos o no hacerle frente a las cargas de trabajo futuras.

La mayor contribución de este artículo es un caso de estudio que presenta la aplicación de varias herramientas y métodos académicos para la exploración de diseños en un ambiente industrial. El caso de estudio presenta cómo se exploran lo cambios en replicación, reasignación(*reallocation* y hardware y su implicación en costo y rendimiento.

Sistema en estudio. Una solución de diagnóstico remoto (RDS por sus siglas en inglés) de la compañía ABB. El sistema RDS tiene más de 150000 líneas de código y es usado para actividades de servicio en miles de dispositivos industriales, fallas y otros datos. ABB desea mejorar el rendimiento de RDS con una nueva arquitectura porque su *back-end* está operando en sus límites de rendimiento y escalabilidad. Las mejoras de corto plazo en el rendimiento y hardware no resolverán de forma sostenible los problemas a largo plazo. La métrica de mayor interés para los *stakeholders* del sistema es el rendimiento de carga(*upload*) del dispositivo, por ejemplo, el número de cargas que el sistema pueda manejar por segundo. Se decidió que el sistema en promedio no deben utilizar más del 50 % de recursos para hacerle frente a las cargas de trabajo pico.

Actividades. 3 principales actividades se llevaron a cabo: medición del rendimiento, modelado del rendimiento y exploración de diseños candidatos.

5.1. Medición del rendimiento

Mediciones son necesarias para crear un modelo de rendimiento preciso. Para medir el rendimiento de RDS de forma precisa, varios paso fueron realizados: primero se seleccionaron las herramientas para la medición, luego se creó un modelo de la carga de trabajo del sistema. Finalmente, se realizaron las mediciones.

5.2. Modelado del rendimiento

Para construir el modelo del rendimiento de RDS, primero se seleccionó una notación de modelaje apropiada que resultó ser el *Palladio Component Model*. Basado en los resultados de las mediciones, el modelo de la carga de trabajo y análisis adicionales, se construyó un modelo de Palladio para el RDS, el cual se calibro4 hasta que reflejará el rendimiento del sistema en estudio.

5.3. Exploración de diseños candidatos

Basado en la modelo del rendimiento creado, se buscó la arquitectura más eficiente en costo para hacerle frente a la carga de trabajo incremental. Se consideraron 3 escenarios y para cada uno de ellos se determinó la arquitectura adecuada que cumpliera con el objetivo de rendimiento. Se ejecutaron varias predicciones manuales para calibrar el modelo. Debido a la gran cantidad de soluciones, se aplicó un herramienta de exploración de diseños candidatos llamada *PerOpteryx*. Se ejecutaron varias predicciones y se creó un plan para aplicar las arquitecturas.

REFERENCIAS

[1] Heiko Koziolek. 2010. *Performance evaluation of component-based software systems: A survey. Perform.* Eval. 67, 8 (August 2010), 634-658. DOI: http://ezproxy.itcr.ac.cr:2075/10.1016/j.peva.2009.07.007

[2] Stephen Becker, Heiko Koziolek and Ralf Reussner. *The Palladio component model for model-driven prediction.* Journal of Systems and Software 82:3-22. Elsevier Science Inc. 2009. DOI: http://dx.doi.org/10.1016/j.jss.2008.03.066

[3] Andreas Brunnert, André van Hoorn, Felix Willnecker, Alexandru Danciu, Wilhelm Hasselbring, Christoph Heger, Nikolas Roman Herbst, Pooyan Jamshidi, Reiner Jung, Jóakim von Kistowski, Anne Koziolek, Johannes Kroß, Simon Spinner, Christian Vögele, Jürgen Walter, and Alexander Wert. *Performance-oriented devops: A research agenda.* Technical Report SPEC-RG-2015-01, SPEC Research Group - DevOps Performance Working Group, Standard Performance Evaluation Corporation (SPEC), 2015. http://arxiv.org/abs/1508.04752

[4] Robert Heinrich, André van Hoorn, Holger Knoche, Fei Li, Lucy Ellen Lwakatare, Claus Pahl, Stefan Schulte, and Johannes Wettinger. *Performance engineering for microservices: Research challenges and directions.* In Companion Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering, 2017, pages 223-226. DOI: https://doi.org/10.1145/3053600.3053653

[5] Thijmen de Gooijer, Anton Jansen, Heiko Koziolek, and Anne Koziolek. 2012. *An industrial case study of performance and cost design space exploration.* In Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE '12). ACM, New York, NY, USA, 205-216. DOI: https://ezproxy.itcr.ac.cr:2878/10.1145/2188286.2188319

A.