Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores

Proyecto de Diseño en Ingeniería en Computadores

Diseño asistido de aplicaciones aproximadas para sistemas computacionales personalizables

Chair for Embedded Systems (CES)
Kalrsruhe Institute of Technology (KIT)

Propuesta de anteproyecto

Daniel Moya Sánchez

9 de marzo de 2018

1. Introducción

Los sistemas de Tecnologías de Información (TI) buscan dar una mejor calidad de vida a las personas. En esta tarea, estos sistemas han tenido que enfrentar ciertos problemas entre los que se puede mencionar el costo en área, potencia y tiempo de ejecución, las cuales son variables que restringuen el rendimiento de un chip. Idealmente, una aplicación debe ajustarse a las necesidades reales del usuario y, en general, del área de aplicación, de forma que se dé un uso óptimo de los recursos. Actualmente, el diseño de procesadores no solo se enfoca en contar con más desempeño si no en tener un manejo de recursos apropiado. No obstante, algunos desafíos en este campo están dados por limitaciones físicas, por ejemplo:

- las características eléctricas de los transistores CMOS, las cuales restringen el consumo de energía en sistemas embebidos y lo cual es un aspecto que deben considerar los diseñadores de componentes para propósito específico en procesadores;
- la pared de memoria, que corresponde a la diferencia entre el crecimiento de la capacidad de procesamiento contra la velocidad de obtención de datos desde memoria;
- y la pared de utilización, la cual limita el uso máximo de hardware simultáneo debido a las capacidades de disipación de calor de un sistema.

Para poder atacar los problemas mencionados anteriormente, una de las áreas de investigación actuales corresponde a la *computación aproximada*, un paradigma de diseño que propone una reducción en la precisión o exactitud de la computación para obtener oportunidades de mejora en cuanto al consumo de área, potencia y tiempo de ejecución. Para aplicar dicho paradigma es necesario identificar aplicaciones tolerantes a errores y determinar, más específicamente, cuáles secciones o funciones dentro de estas pueden ser sustituidas por versiones aproximadas, de forma que se pueda generar un balance entre la calidad de la salida y el consumo general de recursos.

2. Descripción del problema

Al contar con una aplicación que presenta una estructura en *pipeline*, es decir, que posee una serie de etapas donde cada etapa recibe su entrada de una etapa anterior y produce una salida para la etapa siguiente, y donde una o más etapas pueden ser aproximables con más de una versión aproximable (una versión se puede concentrar en mejorar el consumo de potencia, mientras que otra el tiempo de ejecución, por ejemplo) resulta complejo determinar qué combinación de versiones aproximadas utilizar de forma que no se sobrepase el error máximo permitido y a la vez se reduzca, de manera óptima, el uso de ciertos recursos. Dicho proceso podría tomar una cantidad considerable de tiempo si se decide probar todas las posibles combinaciones posibles de versiones aproximadas, por lo que es importante utilizar un esquema de trabajo diferente.

Una aplicación puede tener un comportamiento aproximado si alguna de sus etapas se puede aproximar, ya sea toda una sección o únicamente una instrucción (dentro de una sección). La Figura 1 muestra como ejemplo una aplicación genérica donde ambas situaciones

pueden ocurrir.

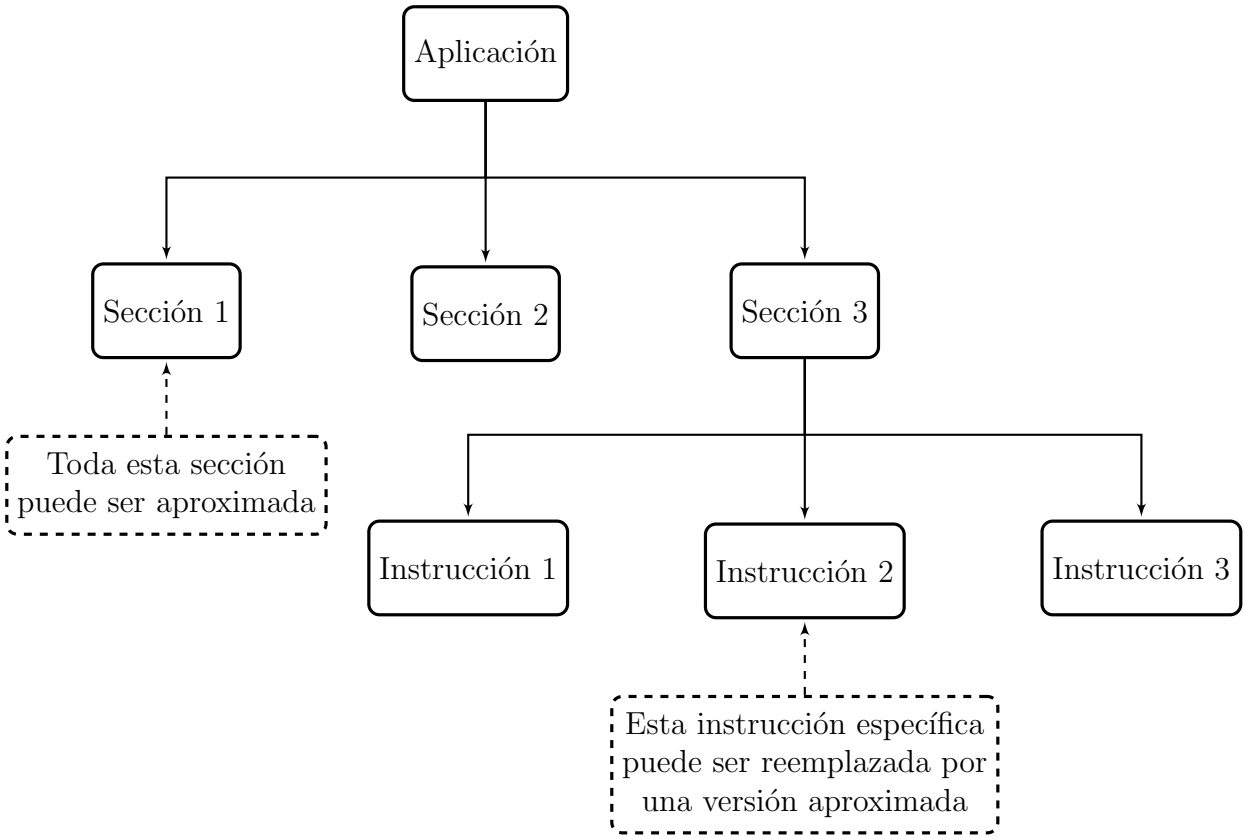

Figura 1: Una posible aplicación aproximada genérica analizada en este proyecto.

Como se muestra en la Figura 1, esta aplicación posee tres secciones, de las cuales la primera (por ejemplo, una etapa de preprocesamiento) puede ser completamente aproximada, la segunda no puede ser aproximada del todo (por ejemplo, una sección crítica de la aplicación) y, finalmente, la tercera tiene tres instrucciones específicas, de las cuales únicamente la segunda posee una versión aproximada.

En el caso hipotético de la Figura 1 donde se tenga más de una versión aproximada para la sección 1 y para la instrucción 2 de la sección 3, es complicado (dicha complejidad aumenta con la cantidad de versiones aproximadas) determinar qué combinación de versiones aproximadas produce la mejor aplicación aproximada final, debido a que, por ejemplo, un cambio en la sección 1 puede impactar severamente las secciones 2 y 3; inclusive, puede que la versión aproximada de la instrucción 2 determine qué tipo de versiones aproximadas son las más convenientes (según las especificaciones del usuario) en la sección 1 para no impactar en gran medida la calidad de la aplicación.

3. Enfoque de la solución

Se busca desarrollar una herramienta de software que pueda escoger entre diferentes versiones para una aplicación aproximada (cada versión dada por una combinación diferente de versiones específicas para cada sección aproximable), según el criterio de usuario que especifique cuáles recursos son críticos en la aplicación y cuál es la cantidad máxima de error permitido. La Figura 2 muestra una abstracción de la implementación de esta herramienta.

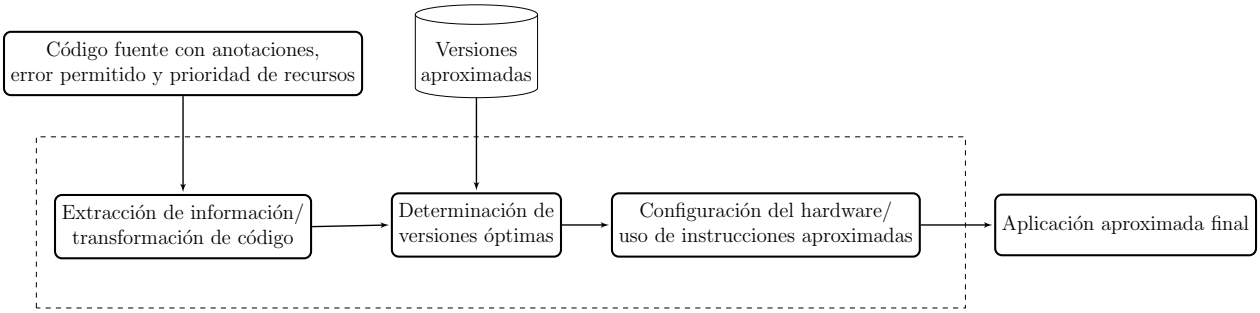

Figura 2: Esquema general de la solución propuesta.

Como se muestra en la Figura 2, se espera recibir un código fuente de una aplicación aproximable, donde previamente el usuario ha indicado, a través de pragmas propios, qué funciones del código son tolerantes a errores; este es transformado a una representación intermedia. El algoritmo que se pretende desarrollar busca poder escoger cuáles versiones aproximadas de las posibles que existen se ajustan para cumplir a cabalidad con el error permitido y, de la mejor manera, con la priorización de recursos. Una vez identificadas las versiones que serán utilizadas, se procederá a utilizar el hardware específico que cuente con el soporte para las instrucciones aproximadas correspondientes, para finalmente entregar una aplicación final aproximada.

3.1. Objetivos

3.1.1. Objetivo General

Desarrollar una herramienta que, a partir de información sobre diferentes versiones de secciones aproximadas de una aplicación tolerante a errores, pueda determinar cuál combinación de versiones genera un mejor resultado en términos de ahorro de recursos y el nivel de error máximo que un usuario estableció como permitido.

3.1.2. Objetivos Específicos

1. Generar una representación del código de entrada que sea manipulable a partir de las anotaciones dadas por un usuario.

2. Evaluar las versiones aproximadas de las secciones de una aplicación para saber cuál es su impacto en la aplicación final.

3. Desarrollar un algoritmo que permita, a partir de funciones indicadas por un usuario, la escogencia de secciones tolerantes a errores en una aplicación según las posibles versiones aproximadas existentes.

4. Validar que la aplicación final aproximada cumpla con el funcionamiento de la aplicación original y las restricciones dadas por el usuario y por el la aplicación

4. Propuesta Metodológica

4.1. Tipificación del trabajo a realizar

El proyecto se clasifica como un trabajo de investigación cualitativa, donde se deben aplicar conocimientos en temas relacionados con sistemas operativos, arquitectura de computadores, computación de alto rendimiento y compiladores.

4.2. Descripción del proceso a realizar

La Figura 3 resume el proceso que se realizará durante el proyecto. Como se puede observar, el proyecto iniciará con una etapa de investigación sobre trabajos realizados por varios autores en el área de computación aproximada, relacionados con la caracterización de sistemas o aplicaciones en las cuales una o varias secciones son aproximables. Se investigará sobre maneras de generar un compilador para un cierto lenguaje y que permita el reconocimiento de pragmas o anotaciones en el código.

Seguidamente, se implementará un algoritmo que, a partir de una información dada (gracias a una base de datos) determine qué combinación de funciones aproximadas se deben colocar en un sistema en pipeline de tal forma que el resultado al final de todas las etapas se mantenga en un nivel de error aceptable. Para esto se tomará información de una base de datos del KIT sobre secciones aproximadas independientes, para posteriormente evaluar el impacto final de cada una de ellas en una aplicación completa.

Finalmente, se debe realizar la verificación de la aplicación aproximada final, de forma que se garantice un cumplimiento en el nivel de error y una optimización apropiada de los recursos. Para esto se realizarán simulaciones y pruebas unitarias en plataformas como ModelSim. Si por alguna razón se detectaran fallas, se revisará el algoritmo desarrollado con el fin de poder corregirlo.

La documentación del proyecto se trabajará a lo largo de todo el proceso de desarrollo, de forma que al final se genere un artículo científico y demás documentos propios de un trabajo final de graduación.

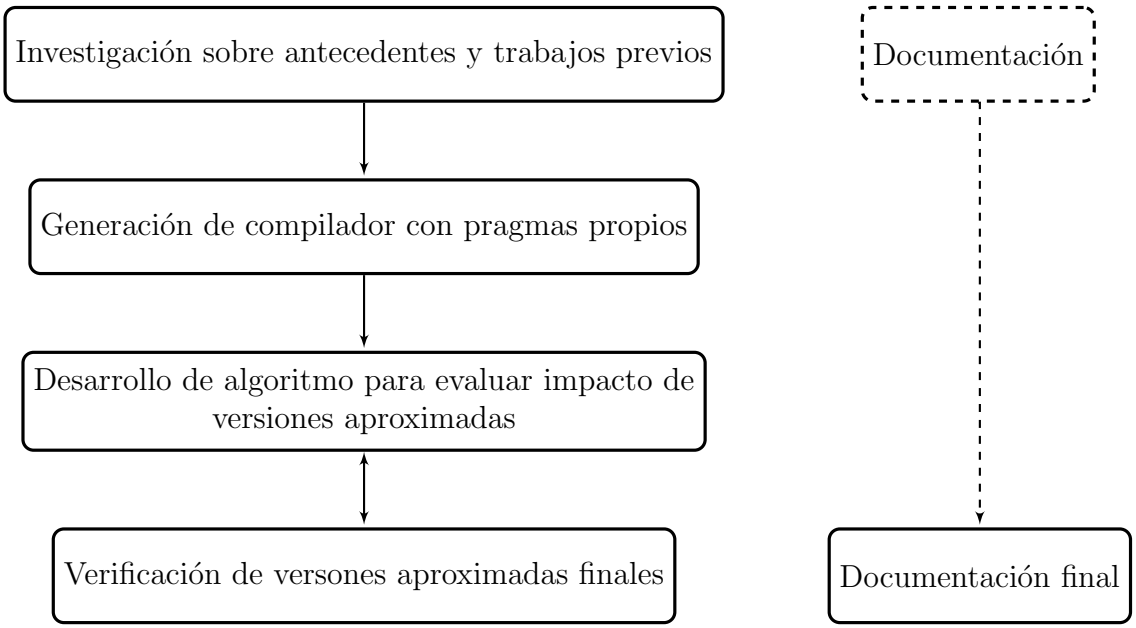

Figura 3: Proceso a realizar en este proyecto.

4.3. Entregables del proyecto

La tabla 1 presenta la asociación entre entregables y objetivos del proyectos, según los apartados mencionados anteriormente.

Cuadro 1: Entregables del proyecto

Objetivo	Entregable
Generar una representación del código de entrada que sea manipulable a partir de las anotaciones dadas por un usuario	Generación de compilador con pragmas propios
Evaluar las versiones aproximadas de las secciones de una aplicación para saber cuál es su impacto en la aplicación final	Documentación y desarrollo del algoritmo
Desarrollar un algoritmo que permita, a partir de funciones indicadas por un usuario, la escogencia de secciones tolerantes a errores en una aplicación según las posibles versiones aproximadas existentes	Desarrollo de algoritmo para evaluar impacto de versiones aproximadas
Validar que la aplicación final aproximada cumpla con el funcionamiento de la aplicación original y las restricciones dadas por el usuario y por el la aplicación	Aplicación aproximada final y documentación

5. Antecedentes y trabajos relacionados

En la actualidad, dada la gran cantidad de aplicaciones complejas (por ejemplo sistemas GPS, reconocimiento de voz, etc.) la computación aproximada ayuda a mantener una salida aceptable mientras se logra que ciertas métricas como tiempo de respuesta o eficiencia

energética se mejoren. En general, la computación aproximada provee la libertad de escoger entre un cierto nivel de error o degradación de la calidad en la salida final de una aplicación (por ejemplo ruido en la señal de la salida) para mejorar el consumo de energía, el área o el tiempo de ejecución; esto sirve como herramienta a un investigador para que ajuste una aplicación dada a las necesidades reales y específicas de esta. En la Figura 4 se muestra un esquema que puede ser aplicado a sistemas tolerantes a errores para incluir en estos la computación aproximada [1].

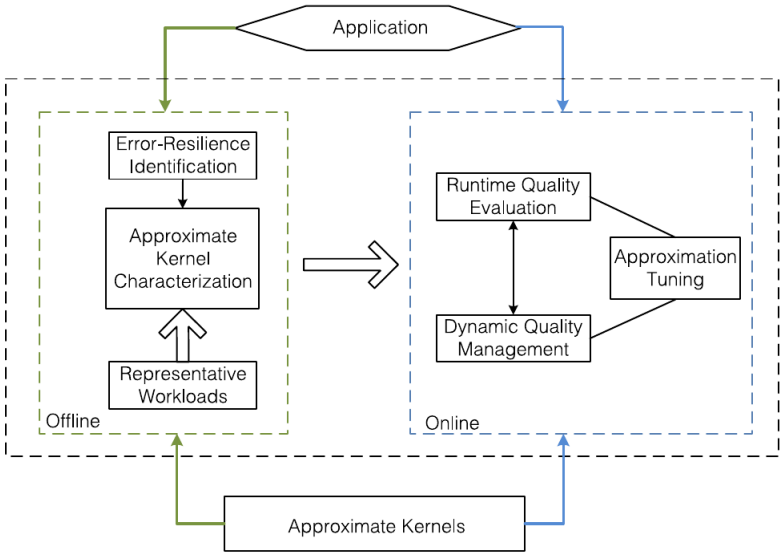

Figura 4: Un marco de trabajo para el uso de computación aproximada. Tomado de [1].

Los elementos clave de la Figura 4 son *kernels* aproximados, los cuales representan la implementación (técnicas) de las funciones aproximadas, estas puede ser realizadas a nivel de hardware o de software; la identificación de la secciones tolerantes a errores y sus características particulares (análisis de impacto); y el manejo de la calidad, el cual implica una evaluación continua para determinar si la aplicación logra los requerimientos deseados.

Como se mencionó, la computación aproximada puede ser implementada tanto a nivel de software como de hardware. En software una implementación típica es a través de *Loop Perforation*, en la cual ciertos ciclos (usualmente con un patrón dado, como por ejemplo las pares) no son computados, lo cual, por ejemplo en una aplicación de cálculo numérico, reduciría la precisión del valor final calculado. A nivel de hardware, se pueden utilizan módulos especializados, por ejemplo aceleradores para programas aproximados utilizando redes neuronales.

El graduado de la carrera Ingeniería en Computadores Juan Carlos Cruz, realizó un trabajo sobre la computación aproximada, donde él se dio la tarea de caracterizar y calendarizar programas tolerantes a errores en una plataforma multi-acelerador. Parte del actual proyecto busca partir de los resultados generados por Cruz, de forma que se pueda utilizar el conocimiento generado sobre secciones ya aproximadas, para poder desarrollar el algoritmo que seleccionará cuál de todas ellas es la mejor según las especificaciones de un usuario.

Referencias

[1] Qiang Xu, Todd Mytkowicz, and Nam Sung Kim. Approximate computing: A survey. *IEEE Design & Test*, 2018.

6. Anexos

6.1. Descripción de la empresa

El Instituto Tecnológico de Karlsruhe (KIT) surge en 2009 a partir de la unión de la Universidad de Karlsruhe, fundada en 1825 como Universidad Fridericiana, y el Centro de Investigación de Karlsruhe. Se ubica en Karlsruhe, en el estado de Baden-Wüttemberg, al suroeste de Alemania.

El Instituto de Ingeniería en Computadores del KIT incluye grupos de trabajo que abarcan los diferentes niveles de abstracción de sistemas computacionales. En el *Chair for Embedded Systems* (CES) se investigan diversos aspectos relacionados con el diseño de sistemas embebidos, desde la confiabilidad de circuitos hasta el manejo de potencia en sistemas multinúcleos.

El presente proyecto será desarrollado en el CES bajo la dirección del M.Sc. Jorge Alberto Castro Godínez, ingeniero en electrónica, investigador y estudiante de doctorado, quien es egresado del Tecnológico de Costa Rica y posee más de dos años y medio como investigador en el Instituto Tecnológico de Karlsruhe.