Sistema de Asignación de Tareas Energéticamente Eficiente en Infraestructuras de Despliegue Variables

Angel Cañete, Mercedes Amor, Lidia Fuentes

[1] Universidad de Málaga
[2] {angelcv, pinilla, lff}@lcc.uma.es

Resumen Cada vez existen más dispositivos de la Internet de las Cosas conectados a Internet que generan una gran cantidad de datos que pueden llegar a congestionar la red en su camino hacia la Nube. Para paliar esta congestión, tecnologías recientes, como el Edge Computing y el Fog Computing, proponen realizar el procesamiento de los datos en dispositivos más cercanos al origen de estos datos. Esto hace que las infraestructuras sobre las que se despliegan las aplicaciones sean cada vez más variables (diferentes tipo de dispositivos, capacidades de cómputo, características de red, etc). En este trabajo se presenta una solución para la asignación óptima de tareas a dispositivos del borde, con el objetivo de minimizar el consumo energético de la ejecución de las aplicaciones. Para ello, utilizamos modelos de variabilidad de Lineas de Producto Software para configurar tanto las aplicaciones como las infraestructuras de despliegue, presentando un modelo general para este último. La configuración de ambas se utiliza como entrada a un marco de trabajo de asignación óptima de tareas, obteniendo como resultado un sistema que proporciona la configuración más eficiente energéticamente en el momento en que el usuario lanza la aplicación, sin comprometer su experiencia como usuario, de forma transparente, escalable, y consiguiendo un importante ahorro energético, como se demuestra en nuestro caso de estudio.

Keywords: Mobile Edge Computing · Mobile Cloud Computing · Eficiencia Energética · Líneas de Producto Software

1. Introducción

La importancia de los sistemas ciber-físicos (CPS) [13] no para de crecer, debido a la masiva popularización de los dispositivos móviles de uso personal, la mejora de la velocidad de comunicación de las redes inalámbricas, el desarrollo de la Internet de las Cosas (IoT) [1] y las bondades de la computación en la Nube (Cloud Computing) [18], que ha jugado un importante papel en el desarrollo y avance de las aplicaciones móviles para la IoT. Se espera que, para el año 2025, existan alrededor de 80 billones de dispositivos de la IoT -desde dispositivos móviles personales, hasta sensores, electrodomésticos y wearables- conectados a

Internet, generando 175 trillones de Gb de datos al año [20]. Estas previsiones implican un aumento de tráfico en la red que no podría ser asumidos por la infraestructura de la Nube. Más recientemente, tecnologías como Edge Computing [21](Computación en el Borde) y Fog Computing [5] llevan las ventajas y el poder de la Nube más cerca de donde se crean, procesan y usan los datos. El uso de estas tecnologías influye en la forma en la cual las aplicaciones se desarrollan y se despliegan para distribuir su ejecución entre los diferentes dispositivos de la IoT, el Edge o el Cloud [22]. En las instituciones y empresas existen dispositivos que están continuamente encendidos, ya sea porque están ofreciendo algún servicio, o simplemente por una mala praxis de los empleados. en cualquier caso, normalmente estos dispositivos disponen de recursos que no consumen o usan. Estos recursos que no están siendo utilizados pueden ser aprovechados por otras aplicaciones, obteniendo un beneficio energético en el proceso.

Por tanto, se necesitan nuevos métodos y herramientas software que ayuden a los desarrolladores de aplicaciones a gestionar la variabilidad del hardware, la complejidad de las infraestructuras de acceso a la red, la distribución de los datos y tomar la decisión de dónde ubicar las diferentes tareas computacionales de un sistema o una aplicación. Estos métodos y herramientas ayudarían a configurar y a desplegar las aplicaciones en diferentes dispositivos, cumpliendo los requisitos de calidad de servicio (QoS) y de usuario.

El objetivo de este trabajo es definir un proceso de asignación de tareas de las aplicaciones que sea capaz de caracterizar y distribuir las tareas más intensivas computacionalmente entre dispositivos del edge, generando configuraciones eficientes energéticamente. Los modelos de variabilidad [3] permiten la configuración tanto de las infraestructuras de despliegue (*Deployment Infrastructure* - DI), en base a los nodos que las conforman, como de las las aplicaciones, en función de las tareas computacionales de las que se componen. La generación de una configuración permite la creación de una estructura de datos con la información necesaria para cumplir con los requisitos funcionales y no funcionales de las aplicaciones. Por último, la información de la estructura de datos sirve de entrada a un marco de trabajo (o *framework*) de asignación óptima de tareas, que determina en qué dispositivo (o nodo) debe desplegarse y ejecutarse cada tarea.

El trabajo se organiza de la siguiente forma: la Sección 2 presenta los antecedentes y los retos del objetivo del trabajo, en la Sección 3 presentamos nuestro enfoque, en la Sección 4 se evalúa nuestra propuesta y, por último, en la Sección 5 damos una conclusión y presentamos el trabajo futuro.

2. Antecedentes y Retos

La descarga de código o carga computacional (*code offloading*) es una técnica comúnmente utilizada para reducir la necesidad de procesamiento en el dispositivo del usuario, distribuyendo la funcionalidad en diferentes dispositivos. Con esta técnica se consigue reducir el tiempo de ejecución, y el consumo energéti-

co, y aumentar o mejorar los servicios que proporcionan los dispositivos de los usuarios. La granularidad de las cargas de trabajo o tareas que se delegan en otros dispositivos a otro es variable: a nivel de método, a nivel de componente o, el más utilizado, a nivel de tareas. El tamaño de las tareas dependerá de la dependencia de los métodos que las conforman.

Tambien se encuentran diferencias según el lugar o distancia en que se encuentran los dispositivos en los que se delegan las tareas. Así, se distinguen las propuestas basadas en la Nube (*Mobile Cloud Computing* o MCC), las basadas en dispositivos cercanos a la frontera o borde de la red *edge* (*Mobile Edge Computing* MEC) y las soluciones híbridas. En las propuestas basadas en MCC, las tareas son delegadas exclusivamente a servidores en la nube [12,14]. En las soluciones basadas en MEC, los dispositivos que conforman las infraestructuras de despliegue (denominados *nodos*) son dispositivos situados o cercanos a la frontera o borde de la red [11,23]. Por último se encuentran las soluciones que aúnan las bondades de ambos paradigmas (MCC + MEC), las denominadas soluciones híbridas [30,24], entre las que se encuentra nuestra propuesta. En este caso, las infraestructuras de despliegue están compuestas tanto por servidores en la nube como por dispositivos del borde.

2.1. Retos

La descarga de tareas tiene asociada una serie de retos que deben ser abordados por todas las propuestas (e.g. restricciones en la latencia de las comunicaciones y en el cómputo, decidir la distribución de carga de trabajo de los nodos, etc). Algunas propuestas de descarga de tareas obvian factores como el tiempo de ejecución de las tareas descargadas al cloud [16], los costes energéticos ajenos a la CPU [31], el tiempo de computación de las tareas en el cloud [7] o el tiempo de envío de la respuesta por parte del cloud, al considerarse ínfimo en comparación con el de cómputo [6]. De la misma forma, existen soluciones basadas en MEC que no tienen en cuenta la congestión de recursos hardware de los dispositivos del edge [16].

En nuestro enfoque, tendremos en cuenta el tiempo y coste energético asociado a la transmisión, procesamiento y respuesta de cada uno de los dispositivos involucrados en la realización de las tareas, así como el estado de carga de los dispositivos del edge. Realizar una asignación óptima de tareas desde el punto de vista energético, teniendo en cuenta estos factores, plantea una serie de retos:

1. **Variabilidad de las infraestructuras de despliegue:** los nodos que conforman la DI tienen diferentes características hardware y software (capacidad de procesamiento, o memoria), y comunicación (interfaces de red, red de acceso, tecnología de comunicación).
2. **Variabilidad de las aplicaciones:** un mismo tipo de aplicación tiene diferentes configuraciones, determinadas por la funcionalidad que se deba llevar a cabo. Cada una de estas configuraciones está compuesta por un conjunto de tareas que se coordinan y comunican entre si.

3. **Cumplir con los requisitos de las aplicaciones:** las tareas que componen las aplicaciones tienen una serie de restricciones que se deben tener en cuenta para cumplir tanto los requisitos funcionales como no funcionales. Entre las necesidades de los requisitos funcionales nos encontramos: dependencias secuenciales entre tareas (e.g., para que comience la tarea *capturaDeImagen* debe terminar la ejecución de la tarea *enfoqueDeCamara*); ciertos requisitos hardware (e.g., la tarea *CapturaDeImagen* requiere ser ejecutada en un dispositivo con cámara); o requisitos software (e.g., la tarea *enfoqueDeCamara* requiere una biblioteca que debe ser instalada en un dispositivo Android). De la misma forma, los requisitos no funcionales implican factores como el tiempo máximo en completar las tareas (e.g., requisitos de QoS).
4. **Asignación óptima de tareas a los dispositivos:** teniendo en cuenta los factores del Reto 3, el problema debe formalizarse y resolverse para realizar una asignación de tareas óptima entre los dispositivos de la infraestructura de despliegue.
5. **Asignación de recursos a usuarios:** se debe comprobar si la infraestructura de despliegue es capaz de asignar la suficiente cantidad de memoria para contener las tareas. En caso afirmativo, se debe optimizar la asignación de estos recursos entre los usuarios.

3. Nuestra propuesta

En esta sección definimos nuestra propuesta para afrontar los retos que el problema plantea. Utilizando como metodología la Ingeniería de Dominio [19], representamos las características de la aplicación y la DI mediante modelos de variabilidad (Retos 1 y 2). Definimos una estructura para gestionar la información de la configuración del modelo de la aplicación y la DI, abordando así el Reto 3. Por último, se formaliza el problema para poder ser resuelto utilizando un SMT solver (Retos 4 y 5). Este proceso queda recogido en la Figura 1.

3.1. Modelado de las DIs y las aplicaciones

Los modelos de variabilidad representan la información de todos los productos posibles para una determinada familia de aplicaciones, modelando sus características y la relación entre ellas. Los modelos de variabilidad se representan como un conjunto de características ordenadas jerárquicamente, compuestas por relaciones padre-hijo y un conjunto de restricciones (*cross-tree constraints*). Estas restricciones suelen ser de inclusión o exclusión (e.g., si la característica A está incluida en la solución, también lo está la característica B).

La representación del problema mediante modelos de variabilidad nos permite plasmar las variables que intervienen en la asignación óptima de tareas. Por un lado, afrontamos el Reto 1 mediante el modelado de la DI. Por otro, abordamos el Reto 2 modelando la familia de aplicaciones.

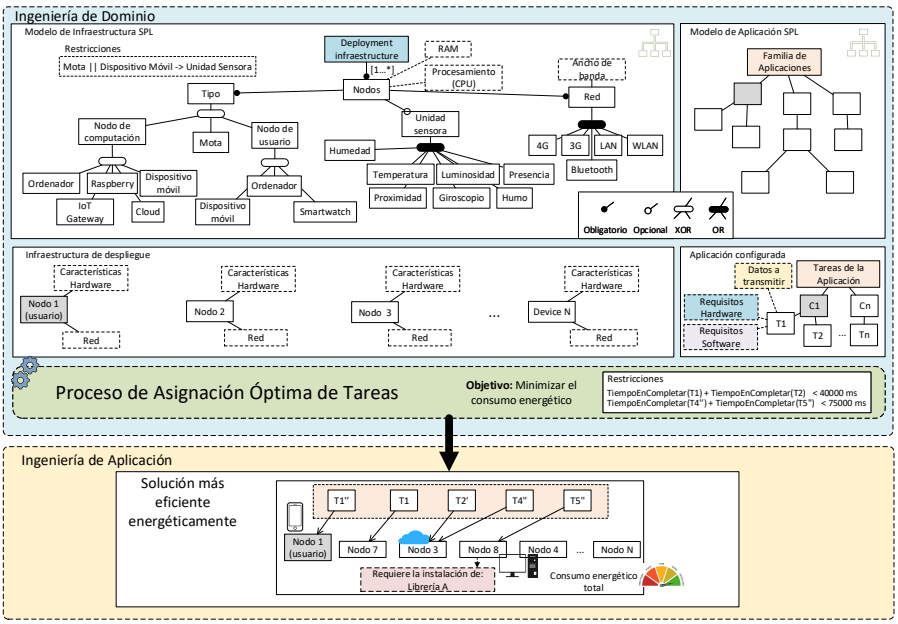

Figura 1. Representación de nuestra propuesta

Modelado de las DIs: las infraestructuras de despliegue están formadas por un conjunto de dispositivos (denominados *nodos*), con una serie de características que los describe: tipo de dispositivo, capacidad de cómputo, cantidad de memoria, conectividad, características software, etc. El modelo de variabilidad de una DI contiene las características necesarias para describir sus nodos. En la parte superior de la Figura 1 podemos ver el modelo de variabilidad de una DI. Cada nodo tiene asignado un tipo, una red a la que están conectados y, opcionalmente, una serie de sensores. El nodo pueden ser de tipo computación (destinado a recibir tareas delegadas por otros usuarios), nodo de usuario (donde la tarea principal se ejecuta) o mota sensora. Entre los nodos destinados a la computación, nos encontramos ordenadores, dispositivos móviles, ordenadores tipo Raspberry o Gateways IoT (*routers* con capacidad de procesamiento) que se encuentran en el edge, así como dispositivos cloud. En nuestro modelo, existe una restricción que obliga al usuario, en el momento de configurar el nodo, a seleccionar un tipo (o tipos) de unidad sensora, en caso de que el nodo que se encuentra configurando sea de tipo mota o dispositivo móvil. Esto nos permitirá definir qué tareas pueden ser ejecutadas en qué nodos, ya que existen tareas que pueden requerir de uno de estos sensores (e.g., medición de temperatura, obtención de una fotografía, etc). Por último, el usuario debe seleccionar la red a la que el nodo se encuentra conectado, que tendrá asociado un ancho de banda. La información sobre la memoria RAM, la capacidad de procesamiento y el ancho de banda se representan mediante el uso de modelos con atributos numéricos. Dado que la DI está formada por uno o varios nodos, utilizamos modelos de variabilidad con cardinalidad [8]. La cardinalidad en un modelo de variabilidad

Parámetros	Definición
τ	Conjunto de todas las tareas de la aplicación
w_i	Número de ciclos de CPU necesarios para llevar a cabo la tarea i
r	Conjunto de tareas con restricción de tiempo $r_i = \{t_1, t_2...t_n\}$
R	Conjunto de todas las restricciones de tiempo de la aplicación $R = \{r_1, r_2...r_n\}$
m_i	Cantidad de memoria RAM requerida por la tarea i
N	Conjunto de todos los nodos de la infraestructura de despliegue
C_n	Capacidad del canal al que se encuentra conectado el nodo n
F_n	Frecuencia de CPU del nodo n
κ_n	Constante del circuito implicada en el consumo energético del nodo n
P_n^{Tx}	Potencia de transmisión del nodo n
$h_{i,j}$	1 si las tareas i y j se ejecutan en dispositivos distintos, 0 de lo contrario
$x_{i,n}$	1 si las tareas i está asignada al nodo n, 0 de lo contrario
$c_{i,j}$	Cantidad de información que la tarea i debe enviarle a j (coste del enlace)
F_n	Capacidad de procesamiento del nodo n
M_n	Memoria RAM de nodo n
U	Conjunto de todos los usuarios que utilizan la aplicación
$m_{n,u}$	Cantidad de memoria RAM asignada al usuario u en el nodo n
$Tproc_{i,n}$	Tiempo empleado en la computación de la tarea i por el nodo n
$Tenv_{n,i,j}$	Tiempo empleado en el envío de información entre la tarea i y j por el nodo n
$Ecomp_n$	Energía consumida por el nodo n en la realización de todas las tareas que tiene asignadas
$Etrans_n$	Energía consumida por el nodo n en el envío de información de todas las tareas que tiene asignadas

Cuadro 1. Variables utilizadas para la formalización del problema

permite definir el número de instancias de una característica, en nuestro caso, el conjunto de los nodos de la DI. Se denota con un intervalo $[n...m]$, siendo n el número mínimo de apariciones de la característica y m el número máximo.

Modelado de la familia de aplicaciones: una familia de aplicaciones está formada por un conjunto de aplicaciones software que comparten la mayoría de sus características. Esto permite configurar y producir no sólo una aplicación en concreto, sino un grupo de ellas, así como la reutilización de componentes software. En los modelos de variabilidad debe incluirse todas aquellas características que definan la aplicación y su comportamiento, de forma que pueda extraerse de este la información necesaria para determinar las tareas que compondrán la aplicación. Por ejemplo, en una aplicación de compresión de vídeo, el mecanismo de compresión o la longitud de la clave son características que formarían parte del modelo, ya que definen su comportamiento.

En nuestro caso, utilizamos la herramienta Pure::Variants [4] para la creación de los modelos de variabilidad. Los resultados de las configuraciones están contenidos en ficheros .xml, de donde se puede extraer fácilmente la información para utilizarla posteriormente en el framework de asignación de tareas.

3.2. Gestión de la información de los modelos

La información obtenida de los modelos de variabilidad debe estar contenida en una estructura de datos que nos permita gestionarla. La estructura propuesta debe representar la información de manera que sea posible asignar las tareas de forma óptima y se cumplan los requisitos de la aplicación (Reto 4 y 5) cumpla los requisitos tanto funcionales como no funcionales de la aplicación. La definición de las variables empleadas se encuentran recogidas en el Cuadro 1.

Para representar las aplicaciones, sus tareas y la conexión entre ellas, utilizamos grafos dirigidos. Esta estructura de datos nos permite mantener la infor-

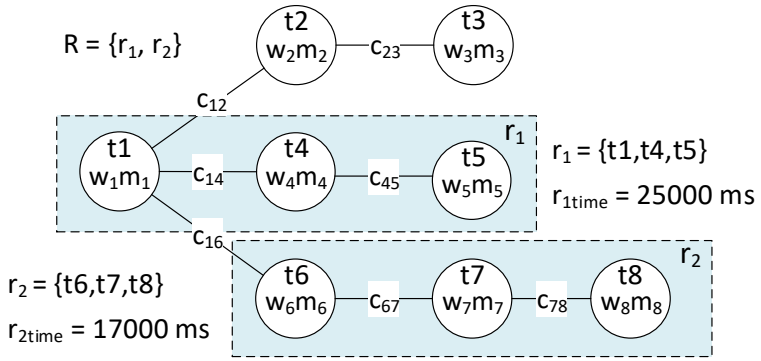

Figura 2. Representación en grafo de las tareas que componen una aplicación.

mación necesaria para cumplir con los requisitos de las aplicaciones (Reto 3). El grafo está compuesto por un conjunto de vértices que representan las tareas τ. Estas tareas contienen tuplas *t(w,m)*, donde (w) contiene la carga computacional asociada, medida en ciclos de CPU, y (m) la cantidad de memoria RAM requerida por la tarea, medida en Mb. Estos valores son comúnmente utilizados para la representación de tareas y pueden ser estimados [17]. El conjunto τ contiene todas las variables de la aplicación. Las aristas del grafo contienen información sobre la cantidad de información que se debe transmitir entre tareas, representada en bits (Figura 2). Si dos tareas i y j no se encuentran conectadas, el peso del enlace será igual a 0. Este tipo de representación nos permite ver que existen tareas cuyo comienzo de ejecución depende de una tarea anterior (dependencia secuencial), como es el caso de la tarea *t1*, *t2* y *t3* en la Figura 2, y la existencia de grupos de tareas paralelizables (por ejemplo, las tareas *t2* y *t4*). Existen conjuntos de tareas con restricciones de tiempo, donde debe cumplirse una condiciones de latencia. Estas tareas son agrupadas en conjuntos de tareas con dependencia secuencial, $r = \{t0, t2, t3..tn\}$, siendo r_{time} el tiempo máximo en ser completadas. El conjunto de estas dependencias componen el conjunto R. Estas restricciones son propias de la aplicación y deben cumplirse para todos los usuarios. En la Figura 2 se muestra el grafo de una aplicación en la que existen dos restricciones de tiempo: en la ejecución de las tareas *t1*, *t4* y *t5*, (r_1) y en la ejecución de las tareas *t6*, *t7* y *t8* (r_2).

La DI está formada por un conjunto de nodos N. Cada nodo tiene una serie de características hardware y de red, representadas como $n(F, M, \kappa, C, P^{Tx})$. F representa la capacidad de procesamiento, medido en ciclos de reloj por segundo (Hz). M es el total de memoria que puede asignar el nodo. κ es una constante que depende del hardware y que influye directamente en el consumo energético de computar una tarea [29]. C (bits/seg) representa la capacidad de transmisión del canal (determinada utilizando la Ley de Shannon [25]) y P^{Tx} (W) el poder de transmisión, que se asume como constante [11]. Estas características de la red a la que se encuentra conectado el nodo determinan la capacidad de transmisión del canal, pudiendo ser calculados utilizando mecanismos de control de energía [26].

La información tanto de la aplicación como de la DI es contenida en el framework de asignación de tareas utilizando estructuras de tipo *Array*. En el caso de la DI, esta información está contenida por completo en la configuración del modelo de variabilidad, por lo que se obtiene directamente. Sin embargo, el modelo de aplicación no contiene información sobre dependencias entre tareas o sus relaciones. En este caso, los valores de los pesos de las tareas y las relaciones entre ellas se encuentran pre-establecidos en el framework por el desarrollador para los posibles valores de entrada de la configuración de la aplicación.

3.3. Formalización del problema

Para formalizar el problema de asignación óptima de tareas, se define el tiempo y consumo energético asociados al procesamiento de las tareas y al envío de la información. Por último, definimos la función de optimización.

Tiempo de procesamiento y envío. El tiempo de procesamiento y envío de información de cada tarea dependerá de sus características, así como del nodo en que se despliegue. El tiempo (seg) de procesamiento de la tarea i por el dispositivo n viene dado por la ecuación:

$$Tproc_{i,n} = x_{i,n}\frac{w_i}{F_n} \tag{1}$$

donde $x_{i,n}$ es 1 si la tarea i se ejecuta en el nodo n, siendo 0 en otro caso.

El tiempo (seg) empleado por el nodo n en enviar la información entre las tareas i y j viene determinado por la ecuación:

$$Tenv_{n,i,j} = x_{i,n}\frac{c_{i,j}}{C_n} \tag{2}$$

Consumo energético de cada nodo. El consumo energético en un nodo está determinado por diferentes factores, como el uso de CPU, almacenamiento o memoria. Dado que el uso de CPU es más influyente que el resto de factores, es la característica que tendremos en cuenta para el consumo energético asociado a la computación de las tareas [22]. El consumo energético de cada nodo (J) de la infraestructura viene determinado por la ecuación [29,10,27]:

$$Ecomp_n = \sum_{i\epsilon\tau} x_{i,n}\kappa_n w_i F_n^2 \tag{3}$$

El consumo energético (J) en un nodo debido las comunicaciones está determinado por [15,11]:

$$Etrans_n = \sum_{(i,j)\epsilon\tau} x_{i,n}h_{i,j}P_n^{Tx}\frac{c_{i,j}}{C_n} \tag{4}$$

donde $h_{i,j}$ es 1 si la tarea i se ejecuta en distinto nodo que la tarea j.

Función de optimización. Una vez determinados los valores que influyen en el tiempo y consumo energético por la computación y envío de información de las tareas acorde con el nodo en el que estas son desplegadas, podemos formalizar el problema de asignación de tareas (Reto 4). La función a minimizar está representada en la Ecuación 5:

$$
\begin{aligned}
&\textbf{Minimizar:} \quad \forall n \epsilon N : Etrans_n + Ecomp_n \\
&\textbf{Sujeto a:} \quad \forall t \epsilon \tau : \sum_{n \epsilon N} x_{t,n} = 1 \qquad (1) \\
&\qquad\qquad \forall n \epsilon N : \sum_{u \epsilon U} (m_{n,u}) \leq M_n \qquad (2) \\
&\qquad\qquad \forall r \epsilon R, n \epsilon N : \sum_{(i,j) \epsilon r} Tproc_{i,n} + Tenv_{n,i,j} \leq r_{\text{time}} \qquad (3)
\end{aligned}
\tag{5}
$$

donde la solución de consumo mínimo de energía está sujeta a una serie de restricciones: la condición (1) comprueba que cada tarea es asignada a un único nodo. La restricción (2) comprueba que la suma de recursos de memoria RAM asignada a los usuarios no supere el total de recursos de memoria de cada nodo (Reto 5). Por último, la condición (3) comprueba que, para cada conjunto de tareas con restricciones de tiempo, la suma del tiempo de cómputo y de envío no supera el tiempo máximo establecido para cumplir con los requisitos de QoS (Reto 3).

Los recursos de memoria disponibles entre los nodos de la DI son repartidos entre los usuarios (U) que hacen uso de la aplicación. Una vez asignados estos recursos por el framework, pueden ser limitados en la práctica de diferentes formas: utilizando un sistema de virtualizanción, sistema de contenedores (por ejemplo, Docker), o mediante el uso de sistemas unikernel.

4. Evaluación

En este apartado se evalúa nuestra propuesta, mostrando los resultados de beneficio de consumo energético para un caso de estudio, así como la escalabilidad de nuestro framework para diferentes tamaños de problema.

Nuestro caso de estudio se encuentra representado en la Figura 3. En la parte superior de esta figura se encuentra el modelo de variabilidad de la familia de aplicaciones (en este caso, la familia de aplicaciones de realidad aumentada, AR [2]), donde el nodo del usuario es un dispositivo móvil. Este modelo contiene la información necesaria para definir qué tareas componen la aplicación. Así, si el usuario desea que la aplicación muestre contenido virtual en función de la ubicación (e.g., Pokémon GO), el entorno aumentado será basado en la ubicación, en este caso obtenida mediante GPS. La configuración de la aplicación determina las tareas que la componen, así como las restricciones de tiempo, requisitos de cómputo y cantidad de información a enviar entre tareas (peso de los vértices y enlaces del grafo de tareas). En el ejemplo mostrado en la parte superior de la

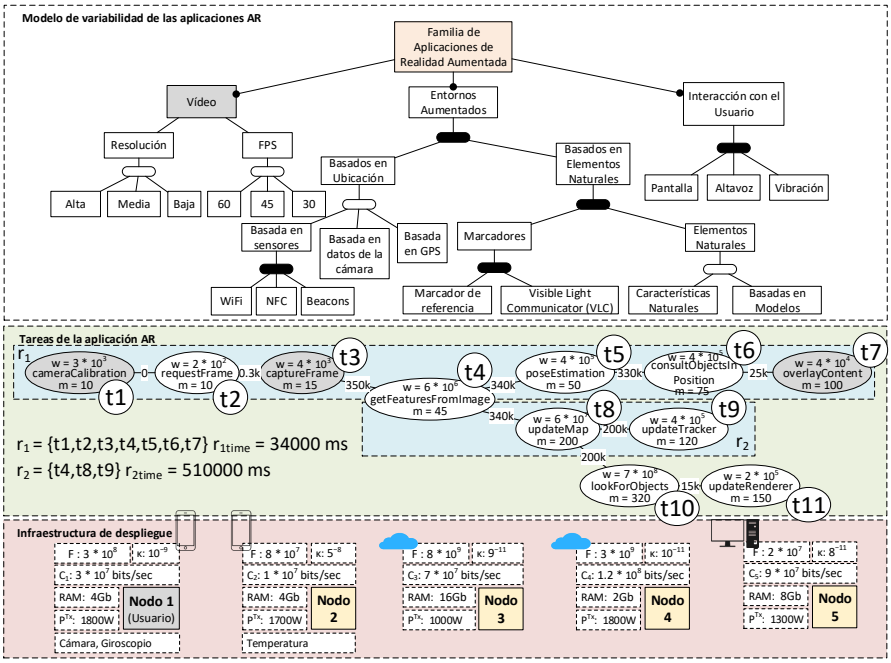

Figura 3. Modelo de la familia de aplicaciones de realidad aumentada.

Figura 3, la resolución seleccionada (alta, media o baja) determina la cantidad de datos que se deben enviar entre las tareas que computan las imágenes, y los FPS (*Frames Per Second*) el tiempo en que las tareas de las restricciones deben ser completas para cumplir los requisitos de QoS.

La configuración del modelo de variabilidad de la aplicación da como resultado el conjunto de tareas mostrado en la parte central de la Figura 3. En el ejemplo mostrado, se ha seleccionado alta resolución de imagen y una tasa de refresco de 30 FPS. Los elementos aumentados están basados en la ubicación (determinada por la cámara) y en elementos naturales (con características naturales). Por último, el contenido de realidad aumentada se muestra al usuario a través de la pantalla y los altavoces. El conjunto de tareas resultante de esta configuración y la relación entre ellas se muestra representada mediante un grafo dirigido, según el esquema introducido al comienzo de la Sección 3.2. Las tareas *t1*, *t3* y *t7* deben asignarse obligatoriamente al nodo del usuario, por cuestiones de dependencias hardware (calibración de la cámara, captura de imagen y superposición de contenido). El resto de las tareas no tienen restricciones funcionales, por lo que podrían ser ejecutadas en cualquier nodo de la DI.

Las configuración de la DI se muestra en la parte inferior de la Figura 3. Esta configuración es el resultado de utilizar el modelo de variabilidad de la parte superior de la Figura 1[3]. En este caso, la DI está compuesta por 5 nodos,

[3] Los valores de las variables utilizados están apoyados en otros trabajos [28,11]

dos de ellos dispositivos móviles (Nodos 1 y 2), donde uno de ellos es el nodo del usuario, dos dispositivos en el cloud (Nodos 3 y 4) y un dispositivo del edge (Nodo 5).

Nodo	**1**	**2**	**3**	**4**	**5**
Tareas	t1,t3,t4,t5 t6,t7,t8	t10,t11	-	-	t2,t9
RAM asignada (Mb)	200	320	-	-	120
Beneficio energético			**58 %**		

Cuadro 2. Asignación óptima de tareas para el problema de la Figura 3

El problema de optimización definido en la Sección 3.3 es modelado utilizando un solver SMT (*Satisfiability Modulo Theories*). En nuestro caso, hemos utilizado el solver Z3 [9], una herramienta de Microsoft de código libre[4]. La implementación del problema da como resultado una aplicación cuyo funcionamiento es proporcionando como microservicio. Este se encuentra activo en uno de los nodos de la DI (nodo manejador) y mantiene información actualizada del estado de cada nodo de la DI (disponibilidad, carga de trabajo, recursos, etc). En este caso, se ejecuta en un nodo con un procesador AMD Ryzen 7 1700X, utilizando un único núcleo.

La asignación de tareas para el problema de la Figura 3, utilizando los valores mostrados en la figura tanto de la configuración de la aplicación como de la DI, se muestra en el Cuadro 2. En este caso, vemos que las tareas *t1, t3, t4, t5, t6, t7* y *t8* han sido asignadas al Nodo 1 (nodo del usuario), reservando 200 Mb de memoria para su ejecución. Las tareas *t10* y *t11* se han asignado al Nodo 2, reservando 320 Mb de memoria, y *t2* y *t9* al Nodo 5, con 120 Mb de memoria reservados. Al resto de nodos de la DI no se les ha asignado ninguna tarea, ya que el framework ha detectado que daría como resultado configuraciones que o bien no superan las restricciones de tiempo, o bien serían más costosas energéticamente. El consumo total de energía estimado para la ejecución de las tareas en los nodos asignados es de $6{,}9 \cdot 10^{17}$J, lo que supone una reducción del consumo del 58 % con respecto a ejecutar todas las tareas en el dispositivo del usuario. Para este problema, el número de asignaciones posibles es $5 \cdot 10^8$. El framework ha dado la solución en un tiempo de 0.76 segundos.

4.1. Escalabilidad de la propuesta

El proceso de asignación de tareas es costoso computacionalmente, ya que el número de posibilidades de asignación es de N^τ. Para medir la escalabilidad de la propuesta, evaluamos el tiempo necesario para obtener una solución del framework para distintos tamaños del problema. Para ello, utilizamos una aplicación *Benchmark* donde seleccionamos el número de nodos de la DI, tareas de

[4] Implementación disponible en: http://caosd.lcc.uma.es/research/rsc/tasks-assignation.zip

Figura 4. Tiempo de ejecución para diferentes configuraciones del problema.

la aplicación y número de usuarios (Figura 4), realizando cada experimento 20 veces y mostrando su media.

Número de tareas: se incrementa el número de tareas a asignar para un único usuario, con un número de nodos fijado en 5. Vemos que una aplicación bastante granulada, donde tenemos 20 tareas diferenciadas, tarda en torno a 1,8 segundos en producir una asignación óptima, lo que es asumible por el usuario en el momento de lanzar su aplicación.

Número de usuarios: por cada usuario, se incrementa el número de tareas en el sistema en N, así como el número de nodos en 1. En este ejemplo, cada vez que un usuario entra en la red, su dispositivo pasa a formar parte de la DI. Esto quiere decir que para el caso de 12 usuarios, existen 16 (12 de ellos nodos de usuario idénticos al Nodo 1 de la Figura 3) nodos en la DI y 132 tareas en total.

Número de nodos: la cantidad de dispositivos que forman parte de la red repercute en el tiempo de ejecución del framework. Para una red de 7 nodos en la DI, el tiempo es de 3,7 segundos, siendo 29 en el caso de 10 nodos (10^{11} posibilidades de asignación).

5. Conclusión

En este trabajo se presenta una solución que modela la variabilidad de las infraestructuras de despliegue de aplicaciones, así como las aplicaciones en sí, utilizando modelos de variabilidad. Los datos de la configuración son recogidos por un framework de asignación óptima de tareas, que devuelve la solución de despliegue más eficiente desde el punto de vista energético, cumpliendo con los requisitos funcionales y no funcionales de las aplicaciones, y asignando una cantidad de recursos en los nodos de la DI a cada usuario. Se ha evaluado un caso de estudio, utilizando la familia de aplicaciones de realidad aumentada, y se muestra un resultado de una reducción en el consumo del 58 % con respecto a la ejecución de todas las tareas en el dispositivo del usuario. Por último, se ha evaluado la escalabilidad de la solución, dando unos resultados que muestran que la asignación puede hacerse en el momento en la aplicación se lanza.

Como trabajo futuro, además de asignar memoria a los usuarios, reservaremos una cantidad de potencia de CPU, convirtiendo el problema de programación

lineal en uno no lineal. Por otro lado, en este trabajo no se considera el nivel de batería del nodo del usuario para definir políticas de despliegue y/o replicación de componentes. Nuestra propuesta evolucionará hacia una solución que tenga en cuenta estos aspectos.

Agradecimientos

Trabajo financiado por los proyectos MAGIC P12-TIC1814, HADAS TIN2015-64841-R (cofinanciado por el fondo FEDER), TASOVA MCIUAEI TIN2017-90644-REDT y MEDEA RTI2018-099213-B-I00 (cofinanciado por el fondo FEDER).

Referencias

1. Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.
2. Ronald T Azuma. A survey of augmented reality. *Presence: Teleoperators & Virtual Environments*, 6(4):355–385, 1997.
3. David Benavides, Sergio Segura, and Antonio Ruiz Cortés. Automated analysis of feature models 20 years later: A literature review. *Inf. Syst.*, 35(6):615–636, 2010.
4. D. Beuche. Modeling and building software product lines with pure::variants. In *2008 12th International Software Product Line Conference*, pages 358–358, Sep. 2008.
5. Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 13–16, New York, NY, USA, 2012. ACM.
6. Huangke Chen, Guipeng Liu, Shu Yin, Xiaocheng Liu, and Dishan Qiu. Erect: Energy-efficient reactive scheduling for real-time tasks in heterogeneous virtualized clouds. *Journal of Computational Science*, 28:416 – 425, 2018.
7. Z. Cheng, P. Li, J. Wang, and S. Guo. Just-in-time code offloading for wearable computing. *IEEE Transactions on Emerging Topics in Computing*, 3(1):74–83, March 2015.
8. Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. Formalizing cardinality-based feature models and their specialization. *Software Process: Improvement and Practice*, 10(1):7–29, 2005.
9. Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
10. Karel De Vogeleer, Gerard Memmi, Pierre Jouvelot, and Fabien Coelho. The energy/frequency convexity rule: Modeling and experimental validation on mobile devices. In Roman Wyrzykowski, Jack Dongarra, Konrad Karczewski, and Jerzy Waśniewski, editors, *Parallel Processing and Applied Mathematics*, pages 793–803, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
11. T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek. Offloading in mobile edge computing: Task allocation and computational frequency scaling. *IEEE Transactions on Communications*, 65(8):3571–3584, Aug 2017.
12. Mads Darø Kristensen and Niels Olof Bouvin. Scheduling and development support in the scavenger cyber foraging system. *Pervasive and Mobile Computing*, 6(6):677 – 692, 2010. Special Issue PerCom 2010.

13. E. A. Lee. Cyber physical systems: Design challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369, May 2008.
14. Y. Lin, E. T. . Chu, Y. Lai, and T. Huang. Time-and-energy-aware computation offloading in handheld devices to coprocessors and clouds. *IEEE Systems Journal*, 9(2):393–405, June 2015.
15. S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi. Optimal joint scheduling and cloud offloading for mobile applications. *IEEE Transactions on Cloud Computing*, pages 1–1, 2018.
16. Y. Mao, J. Zhang, and K. B. Letaief. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, 34(12):3590–3605, Dec 2016.
17. S. Melendez and M. P. McGarry. Computation offloading decisions for reducing completion time. In *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 160–164, Jan 2017.
18. Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.
19. Klaus Pohl, Günter Böckle, and Frank J van Der Linden. *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media, 2005.
20. Rydning Reinsel, Gantz. The digitalization of the world: From edge to core. 2018.
21. W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, Oct 2016.
22. T. X. Tran and D. Pompili. Joint task offloading and resource allocation for multi-server mobile-edge computing networks. *IEEE Transactions on Vehicular Technology*, 68(1):856–868, Jan 2019.
23. Tim Verbelen, Tim Stevens, Pieter Simoens, Filip De Turck, and Bart Dhoedt. Dynamic deployment and quality adaptation for mobile augmented reality applications. *Journal of Systems and Software*, 84(11):1871 – 1882, 2011. Mobile Applications: Status and Trends.
24. S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung. Dynamic service migration in mobile edge-clouds. In *2015 IFIP Networking Conference (IFIP Networking)*, pages 1–9, May 2015.
25. A. Wyner. Recent results in the shannon theory. *IEEE Transactions on Information Theory*, 20(1):2–10, January 1974.
26. Mingbo Xiao, Ness B. Shroff, and Edwin K. P. Chong. A utility-based power-control scheme in wireless cellular systems. *IEEE/ACM Trans. Netw.*, 11(2):210–221, April 2003.
27. Wanghong Yuan and Klara Nahrstedt. Energy-efficient cpu scheduling for multimedia applications. *ACM Trans. Comput. Syst.*, 24(3):292–331, August 2006.
28. W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu. Energy-optimal mobile cloud computing under stochastic wireless channel. *IEEE Transactions on Wireless Communications*, 12(9):4569–4581, Sep. 2013.
29. W. Zhang, Y. Wen, and D. O. Wu. Energy-efficient scheduling policy for collaborative execution in mobile cloud computing. In *2013 Proceedings IEEE INFOCOM*, pages 190–194, April 2013.
30. Tianchu Zhao, Sheng Zhou, Xueying Guo, Yun Zhao, and Zhisheng Niu. A cooperative scheduling scheme of local cloud and internet cloud for delay-aware mobile cloud computing. 11 2015.
31. X. Zhu, L. T. Yang, H. Chen, J. Wang, S. Yin, and X. Liu. Real-time tasks oriented energy-aware scheduling in virtualized clouds. *IEEE Transactions on Cloud Computing*, 2(2):168–180, April 2014.