Universidad Zaragoza
1542

Trabajo Fin de Grado

Análisis y optimización del coste de utilizar recursos de computación en la nube de Amazon mediante instancias puntuales

Autor/es

Rubén Sanz Sanz

Director
Sergio Hernández de Mesa

Ponente
Francisco Javier Fabra Caro

Universidad de Zaragoza
Curso 2014/2015

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. RUBÉN SANZ SANZ,

con nº de DNI 17770077R en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) GRADO, (Título del Trabajo) ANÁLISIS Y OPTIMIZACIÓN DEL COSTE DE UTILIZAR RECURSOS DE COMPUTACIÓN EN LA NUBE DE AMAZON MEDIANTE INSTANCIAS PUNTUALES,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 20 de Noviembre de 2015

Fdo:

RESUMEN
Análisis y optimización del coste de utilizar recursos de computación en la nube de Amazon mediante instancias puntuales

El *cloud computing* (computación en la nube) ha surgido como un paradigma de computación adecuado para la resolución de aplicaciones complejas. Dentro de este modelo, se ofertan varios tipos de instancias entre las que se encuentran las instancias puntuales (*spot instances*). Las instancias puntuales funcionan mediante un sistema de pujas donde los recursos que se asignan pueden perderse en cualquier momento. Se trata del modelo de instancia más económico y está dirigido a la ejecución de tareas de corta duración.

En este proyecto se plantea el uso de las instancias puntuales para ejecutar tareas científicas. Se ha tratado de estudiar el comportamiento al que están sujetos este tipo de instancias y la viabilidad de su uso (dado su carácter más económico) intentado dar una solución a las múltiples restricciones y variaciones que presentan a la hora de utilizarlas de forma fiable.

Para ello se ha creado un sistema que permite tanto analizar el coste histórico de este tipo de instancias como analizar la viabilidad de ejecutar tareas que deben cumplir un *deadline* sin exceder un presupuesto dado. El sistema está formado por tres componentes. En primer lugar, un componente para obtener de forma automática los históricos de precios de las instancias. En segundo lugar, un sistema de predicciones que para una tarea concreta con unos parámetros fijados (presupuesto, *deadline*, tipo de máquina, región, etc.) da información sobre qué zona es la idónea y en qué medida. Para la predicción, se ha propuesto una nueva métrica basándose en si la tarea podría haberse ejecutado satisfactoriamente en función de datos históricos. Finalmente, un simulador que permite comprobar la eficacia de las predicciones simulando el resultado de ejecutar una tarea con unas condiciones dadas (zona y precio de puja).

La realización de una serie de pruebas en diferentes condiciones nos ha permitido evaluar la utilidad de las herramientas desarrolladas. Así hemos sido capaces de demostrar las ventajas de usar instancias puntuales en cuanto a reducción de coste y hemos detectado situaciones en las que los resultados del predictor y del simulador concuerdan y otras situaciones contrarias que nos han permitido definir interesantes líneas de trabajo futuro.

Tras las pruebas realizadas con el predictor y el simulador se puede establecer que las instancias puntuales, a pesar de lo aleatorias que pueden resultar en algunos casos, son adecuadas para tareas científicas de corta duración. Basándonos en los resultados, se puede asegurar que las instancias puntuales consiguen un alto porcentaje de éxito (más del 85%) en tareas de hasta 10 horas de duración, con un coste varias veces inferior al de las instancias bajo demanda. Consideramos los resultados muy positivos al tratarse de un planteamiento totalmente genérico y adaptable a los cambios que pueda sufrir el modelo de instancias puntuales de Amazon.

Contenido

1. Introducción

En este capítulo se introducirán brevemente la motivación y el contexto de este trabajo de fin de grado. Además, se indicarán los objetivos del mismo y la organización del resto de la memoria.

1.1. Motivación

En computación cada vez se plantean algoritmos más complejos que requieren una gran capacidad de procesamiento como puede suceder en disciplinas como por ejemplo la bioinformática, el procesamiento de imágenes y todo lo relacionado con el *Big Data* [B10, B11]. Para reducir el tiempo necesario en cálculos complejos ya existen soluciones basadas en computación distribuida como es el caso de los *clusters*, *grids* y supercomputadores [B12, B13]. Si se desea optar a soluciones de este tipo es necesario hacer un desembolso de dinero considerable en equipos, infraestructura, personal y mantenimiento entre otros [B14, B15]. Además, este tipo de infraestructuras disponen de unos recursos limitados y son compartidas por una gran cantidad de usuarios. Por otro lado, como alternativa, han surgido las soluciones de computación en la nube (*cloud computing*) donde podemos alquilar dicha potencia de procesamiento por el tiempo que creamos oportuno pagando únicamente por los recursos utilizados y evitando, por tanto, algunos de los costes anteriormente citados [B16, B17].

La aparición del paradigma de computación en la nube (*cloud computing*) ha propiciado un cambio en el modo de operar otorgando la posibilidad de manejar un entorno flexible sin las desventajas que puede suponer operar con un *cluster* o supercomputador [B18]. Esta adaptabilidad a las necesidades de cada usuario queda patente en la variedad de servicios ofertados actualmente. Sin necesidad de realizar una inversión en hardware con todo lo que ello conlleva (mantenimiento, gasto eléctrico, renovación cada X años) se puede acceder entornos completamente distintos únicamente el tiempo que nos es necesario suponiendo un ahorro en costes realmente importante [B19, B20, B21]. Esta adaptabilidad queda patente por los tres diferentes modelos de *cloud* existentes, que se explican a continuación.

En primer lugar tenemos el *IaaS* (*Infrastructure as a Service*), también llamado *HaaS* (*Hardware as a Service*). En este modelo la infraestructura se ofrece como servicio usando para ello (por lo general) una plataforma de virtualización. De este modo el usuario no necesita comprar todos los recursos hardware y los contrata directamente a un proveedor externo pagando únicamente por los recursos utilizados y ahorrando los costes derivados de la compra, mantenimiento y utilización de recursos hardware. Ejemplo: Azure[W14], Google Cloud Platform[W17], Amazon EC2[W10].

Continuamos con el *PaaS* (*Platform as a Service*). Se oferta una encapsulación de servicios básicos (una plataforma) que puedan soportar la integración con una tecnología concreta de desarrollo. El usuario puede controlar las herramientas pero no la infraestructura que las soporta. Ejemplo: Google APP Engine[W15]

Por último tenemos el *SaaS* (*Software as a Service*). Una aplicación se ofrece como servicio y múltiples clientes la consumen a través de Internet (como por ejemplo mediante un navegador web). Ejemplo: Office365[W16], Google Docs[W18].

En este proyecto estamos interesados en el uso de un *IaaS* ya que el servicio contratado son recursos hardware (independientemente de que puedan ser virtualizados) y tenemos control sobre ellos. Estos recursos pueden ser entonces utilizados para la ejecución de aplicaciones científicas [B22, B23]. Haciendo uso de un *IaaS* se nos brinda diversas ventajas donde la principal es el coste inmediato de la inversión a realizar. Por otro lado se dispone de una versatilidad absoluta ya que se pueden comprar instancias de multitud de tipos cada una mejor para según qué tipo de cálculos o tareas, no se está atado a un único hardware. Estos dos aspectos hacen que el entorno *cloud* pueda ser de utilidad en el ámbito científico donde la experimentación y los bajos presupuestos son la norma hoy en día.

Durante este proyecto se va a utilizar la infraestructura de computación en la nube de Amazon [W8]. Concretamente, utilizaremos su servicio de *IaaS*, Amazon EC2 [W10]. "*Amazon Elastic Compute Cloud* (Amazon EC2) es un servicio web que proporciona capacidad informática con tamaño modificable en la nube. Está diseñado para facilitar a los desarrolladores recursos informáticos escalables basados en web." La decisión de utilizar este entorno en concreto se debe a que se trata de la infraestructura *cloud* más utilizada en la actualidad, tanto a nivel académico como industrial, y a que proporciona interesantes modelos de coste que pueden ser utilizados para la ejecución de aplicaciones científicas posibilitando un elevado ahorro de costes.

A continuación describimos las características fundamentales de los diferentes modelos de coste ofrecidos dentro del *cloud* de Amazon.

1. Instancias bajo demanda: En las instancias bajo demanda el usuario paga por cada hora de uso en función de la capacidad contratada. Puede ampliar o disminuir la capacidad del servicio contratado. En este tipo de instancias el usuario no está obligado a ningún tipo de permanencia por lo que en cualquier momento puede interrumpir el proceso (no es necesario hacer contrataciones de 1 día entero si sólo se necesitan 3 horas, por ejemplo). Son el modelo estándar de instancias.

2. Instancias reservadas: Las instancias reservadas funcionan de modo similar a las instancias bajo demanda. Su principal diferencia reside en el tiempo que es necesario contratar cada instancia (1 o 3 años) y que el precio a pagar es independiente de que el usuario utilice o no los recursos durante el tiempo contratado. La ventaja de contratar durante un largo periodo de tiempo es la disminución del precio en comparación con utilizar los mismos recursos con el modelo bajo demanda durante un largo período de tiempo.

3. Instancias puntuales: Las instancias puntuales, al igual que las instancias bajo demanda, se pagan por horas. Los precios de estas instancias, al contrario que en los otros dos tipos, son variables. La compra se realiza mediante un sistema de pujas y si la el precio de la instancia (variable) aumenta por encima del de la puja se pierde el control de los recursos. Estas diferencias se explican más en profundidad en otros puntos de la memoria.

Si bien la propia naturaleza de la computación en la nube puede llevar a un ahorro de coste, gracias a la reducción de los costes de adquisición y mantenimiento de los recursos, en la actualidad se está intentando ir un paso más allá en el aprovechamiento de los recursos económicos haciendo uso de los modelos de coste alternativos proporcionados por los proveedores de *cloud*. En el caso de Amazon tanto la utilización de las instancias reservadas como de las puntuales es de interés en este aspecto. Las instancias reservadas resultan ideales para llevar a cabo proyectos de larga duración, mientras que las instancias puntuales son más adecuadas para ejecutar tareas de corta duración.

En este proyecto, estamos interesados en el aprovechamiento de las instancias puntuales ya que consideramos que son ideales para ejecutar computaciones científicas de corta duración y para lidiar con picos de carga que no pueden ejecutarse en *clusters*, *grids* o supercomputadores cuando estas infraestructuras están sobrecargadas y las tareas deben ejecutarse antes de un determinado plazo (*deadline*).

Numerosas publicaciones han abordado el tema de aprovechar las instancias puntuales ante el interés que podría tener para áreas como la investigación científica donde no siempre se disponen de recursos económicos suficientes [B1-B9]. Las publicaciones anteriores plantean diferentes métodos para el aprovechamiento de las instancias puntuales. Independientemente del proceso seguido y los resultados obtenidos en ellas, todas tienen un denominador común, intentan modelar de manera más o menos formal la lógica del sistema de Amazon a la hora de establecer precios con el objetivo de obtener un modelo que permita determinar el mejor precio para garantizar la máxima disponibilidad y el mejor precio. Sin embargo, las propuestas

anteriores se han encontrado con la dificultad de obtener un modelo adecuado [B1] y con el hecho de que Amazon varía su política de forma periódica [B2].

Como alternativa a propuestas anteriores, en este proyecto, se propone un sistema capaz de evaluar si la ejecución de una tarea utilizando instancias puntuales es o no viable de acuerdo a los resultados históricos de los últimos meses. De esta forma, se intenta proporcionar una solución genérica que no dependa de la lógica utilizada por Amazon para fijar los precios. Con este enfoque, pretendemos estudiar en qué medida un sistema más simple puede ser efectivo a la hora de evaluar la viabilidad de ejecutar tareas usando instancias puntuales. Además, el análisis de la viabilidad presentado por el sistema permite al usuario optimizar tanto el coste como la disponibilidad mediante el análisis de diferentes precios de puja.

1.2. Objetivos

Los objetivos del proyecto son los siguientes:

1. **Analizar el funcionamiento de las instancias puntuales**:
   Estudiar en qué consiste el modelo que oferta Amazon, qué ventajas y desventajas tiene y varias consideraciones que deben tenerse en cuenta a la hora de trabajar con ellas.

2. **Valorar los estudios realizados hasta el momento y extraer conclusiones:**
   Conocer el trabajo y conclusiones a las que han llegado otras personas con el fin de obtener una mejor visión del conjunto y aprovechar las consideraciones que plasmaron en ellos y que a priori podrían no ser fácilmente deducibles.

3. **Situar el uso de las instancias puntuales en un contexto de computación científica:**
   Establecer en qué escenarios puede tener sentido su uso basándonos en sus funcionalidades y limitaciones. Definir las características de los trabajos que pueden ser ejecutados en este tipo de entorno.

4. **Crear una aplicación que gestione su uso:**
   Desarrollar un conjunto de herramientas que determinen si una tarea es viable dentro de una configuración y presupuesto. Dar información fácilmente interpretable al usuario para la ayuda a la toma de decisiones.

5. **Concluir si es un modelo que podría funcionar y bajo qué circunstancias:**
   Valorar los resultados obtenidos en las pruebas realizadas con el modelo que se haya establecido y determinar si su uso resultaría viable y en qué escenarios y circunstancias.

1.3. Organización de la memoria

Capítulo 2: Introducción y análisis de las instancias puntuales. Se estudian los resultados obtenidos en otros trabajos y se llegan a una serie de conclusiones a partir de las cuales se empieza a modelar el sistema a diseñar.

Capítulo 3: Diseño de los componentes de la solución y explicación sobre cómo se ha planteado el cálculo de viabilidad de una tarea.

Capítulo 4: Implementación de la solución, tecnologías utilizadas y diseño de la base de datos.

Capítulo 5: Verificación del sistema. Realización de pruebas de cada componente en varios escenarios y pruebas reales para realizar una comparativa.

Capítulo 6: Conclusiones finales, posibles vías de trabajo futuro y valoración personal del proyecto.

Anexo I: Se detallan más aspectos de la implementación que no se incluyeron en la memoria principal.

Anexo II: Manual de uso para la instalación y uso de algunos de los componente que se utilizaron a lo largo del proyecto.

Anexo III: Generación de gráficas con Gnuplot a partir de los históricos de precios de las instancias puntuales.

Anexo IV: Gestión del proyecto y metodología seguida.

Bibliografía: Referencias bibliográficas que aparecen a lo largo de la memoria.

2. Análisis

En esta sección se introducen las instancias puntuales presentando ciertas consideraciones de importancia para comprender su funcionamiento y hacer un uso coherente de ellas. Para ello, se estudiaron diversos trabajos.

Basándonos en dicho análisis, se plantean los componentes necesarios para un sistema que sea capaz de aprovechar este tipo de instancia.

2.1. Análisis de las instancias puntuales

Desde 2009 las instancias puntuales ofrecidas por Amazon [W10] han despertado el interés de investigadores de diversa índole en un afán por comprender su comportamiento y poder establecer un modelo que pueda ayudar a conseguir un ahorro de costes significativo sin renunciar a la fiabilidad. En este apartado analizamos sus características fundamentales.

2.1.1. ¿Qué es una *spot instance*?

Las instancias puntuales poseen las mismas características (*hardware* y *software*) que las instancias bajo demanda pero están sujetas a un modelo económico distinto y por lo tanto el tratamiento que deben recibir es totalmente diferente. La principal ventaja de las instancias puntuales es su bajo coste, siendo bastante inferior al de las instancias bajo demanda.

A continuación se presenta la tabla 1 con algunas comparaciones de precios entre instancias bajo demanda y puntuales extraídas de la propia web de Amazon (Todas de Estados Unidos y Linux/Unix). En ella puede observarse cómo los precios de las instancias puntuales son considerablemente más bajos que en las instancias bajo demanda.

Tipo de instancia	Precio bajo demanda	Precio puntual
m3.medium	$0.096 por hora	$0.0099 por hora
m3.large	$0.193 por hora	$0.0171 por hora
c3.8xlarge	$2.043 por hora	$0.3075 por hora
g2.2xlarge	$0.898 por hora	$0.0720 por hora

Tabla 1: Comparativa de precios de instancias bajo demanda y puntuales con sistema Linux/UNIX en la región de Asia Pacífico (Tokio) a 1 de Nov de 2015

Comprar una instancia puntual no se hace del mismo modo que una instancia bajo demanda. El precio de las instancias puntuales varía a lo largo del tiempo de acuerdo a la oferta y la demanda. En las instancias puntuales el usuario realiza una puja por un recurso. Si el precio de puja supera al precio de la instancia, Amazon le asigna los

recursos por los que ha pujado. Si por el contrario la puja no supera el precio actual, la puja se mantiene hasta que supere el precio, este caso puede darse 1 minuto después o nunca. Del mismo modo, puedes perder una instancia si el precio de la misma supera la puja realizada. En este caso se pierden los recursos y Amazon no cobra la última hora de utilización de los mismos.

Amazon cobra el precio de la instancia cuando los recursos son asignados al usuario y no al inicio de puja. La variación de los precios es controlada por Amazon de forma privada y por tanto desconocida, por lo que los eventos de asignación y desasignación de recursos no pueden ser controlados 100%.

2.1.2. Trabajo relacionado

En esta sección, se van a describir los aspectos más importantes de los trabajos más relevantes relacionados con instancias puntuales. El objetivo es identificar tanto los problemas y retos que introducen este tipo de instancias así como las soluciones que otros autores han planteado a los mismos. De dichos trabajos se pueden establecer unas pautas iniciales para posteriormente plantear un sistema que se adecue a la realidad del comportamiento de las instancias ofertadas por Amazon.

En primer lugar, analizamos diferentes artículos que estudian el comportamiento de las instancias puntuales [B1, B2, B3].

En el artículo [B1] se analizan diferentes tipos de instancias puntuales y se intenta determinar la dinámica de la variación de los precios a lo largo del día y de la semana. Esta dinámica puede ser modelada por una serie de distribuciones gaussianas con un buen grado de precisión aunque no en todos los tipos de instancia. Otro resultado interesante encontrado los por los autores es que se dan diferentes ciclos en los precios en función de la hora del día y del día de la semana.

En otras publicaciones como [B2] se intenta conocer cómo Amazon establece los precios de sus instancias mediante ingeniería inversa. Pretenden establecer un modelo que genere precios de forma consistente basándose en históricos de precios. A pesar de poder aproximar cómo podrían funcionar los precios concluyen que hay un componente que funciona de forma aleatoria y resulta imprevisible.

Otros como Sewook Wee optan en [B3] por dar un planteamiento totalmente distinto y basar el ahorro en movilizar el trabajo de una instancia a otra en caso de que el precio sea menor o en realizar las tareas en horas de baja carga en lugar de en las horas pico. Estimaron un ahorro de poco más del 3% por lo que no merece la pena llevarlo a cabo.

Tras los estudios aportados se puede concluir que a pesar de poder realizar aproximaciones para entender el funcionamiento de la variación de los precios no siempre sirven para todos los escenarios y tipos de instancias.

Por otro lado, estudiamos algunos de los trabajos más relevantes en cuanto a técnicas para disminuir el coste y seleccionar el precio de puja al utilizar instancias puntuales [B4, B5] donde estudian el comportamiento de los precios de las instancias a lo largo del tiempo y tratan de establecer un modelo matemático de predicción para ajustar las pujas al máximo posible de forma dinámica.

En [B4] tratan el problema en un escenario simplificado donde obtienen buenos resultados. Se centran en un tipo de instancia y no entra en juego el *deadline* o fecha límite para la ejecución de una tarea. En [B5] incorporan el *deadline* en el estudio y tratan de reajustar las pujas para que las tareas se cumplan dentro de los límites impuestos. En ambos casos parten de los *logs* de los precios de las instancias que facilita Amazon.

2.1.3. Regiones y zonas geográficas

Amazon dispone de varias regiones localizadas en diferentes puntos geográficos. Cada una de las regiones corresponde con un *cloud* completamente independiente de los otros. A continuación se listan las regiones existentes en el momento del desarrollo de esta memoria:

- EE.UU. Este (Norte de Virginia)
- EE.UU. Oeste (Oregón)
- EE.UU. Oeste (Norte de California)
- UE (Irlanda)
- UE (Frankfurt)
- Asia Pacífico (Singapur)
- Asia Pacífico (Tokio)
- Asia Pacífico (Sídney)
- América del Sur (Sao Paulo)
- AWS GovCloud (EE.UU.)

Cada una de estas regiones dispone de varias zonas donde oferta instancias puntuales. Cada zona es independiente del resto y el usuario puede seleccionar una zona concreta para sus cómputos o puede dejar que Amazon seleccione una de ellas de forma automática.

Amazon deja a disposición de los usuarios el histórico de precios más reciente (2 meses) al que puede accederse desde la web [W7] o bien mediante sus servicios web [W8] (opción utilizada y explicada más adelante en la memoria).

En cada una de ellas los precios varían con respecto a las otras y una opción que a priori pudiese resultar apetecible por el precio que tuviese en ese momento puede cambiar radicalmente ante sucesos inesperados (por ejemplo: un aumento repentino de la demanda del tipo de instancia que estemos usando)

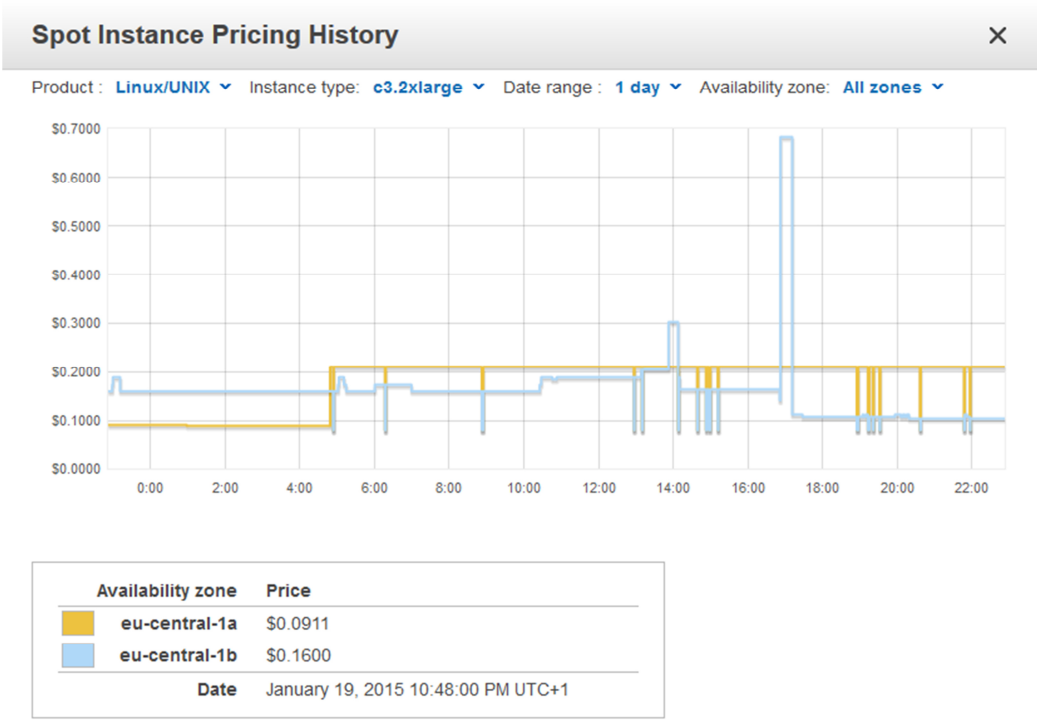


Figura 1: Diferencias de precio entre zonas

Como puede observarse en la figura 1 [W7], para el caso de *eu-central* (que corresponde a la región de Frankfurt), hay disponibles dos zonas. En cada una de ellas el precio varía de una forma diferente. En estos momentos se ve la complejidad que todo conlleva ya que incluso comparando máquinas iguales de una misma región las diferencias entre zonas pueden ser bastante grandes y con comportamientos muy variables. Por tanto, no podemos asegurar una predicción con un 100% de fiabilidad al no conocer el sistema en su totalidad y depender de variables externas que no podemos controlar. Sin embargo, podemos beneficiarnos de este hecho para seleccionar la zona más adecuada en cada momento y optimizar tanto el coste como las probabilidades de que una tarea sea ejecutada sin que la instancia sea desasignada.

A pesar de la aleatoriedad del sistema se pueden extraer algunas conclusiones. Así pues, se observó que las instancias puntuales son un 50% más baratas que sus homónimas bajo demanda (de media). Como se ha comentado, los precios varían dependiendo de las zonas dentro de una misma región. Asimismo, los precios varían dependiendo de la zona, por lo que también podría usarse este hecho para reducir el coste.

2.1.4. Relación entre el precio y el instante temporal

También es interesante destacar el descenso de precios en fines de semana o a partir del mediodía [B1]. Dichos espacios temporales son además los que menos actividad deberían registrar al no corresponderse (por norma general) a jornadas laborales. Por lo tanto podemos deducir que el precio de las instancias está relacionado de algún modo con la demanda que haya de éstas.

En las figuras 2 y 3 se muestran las variaciones de los precios en función de la hora del día y del día de la semana. Se observa una tendencia similar en un variado conjunto de tipos de instancias. Las bajadas se deben a la menor demanda de instancias, es por ello que de 3 a 9 de la tarde y los fines de semana los precios de las instancias sean menores.

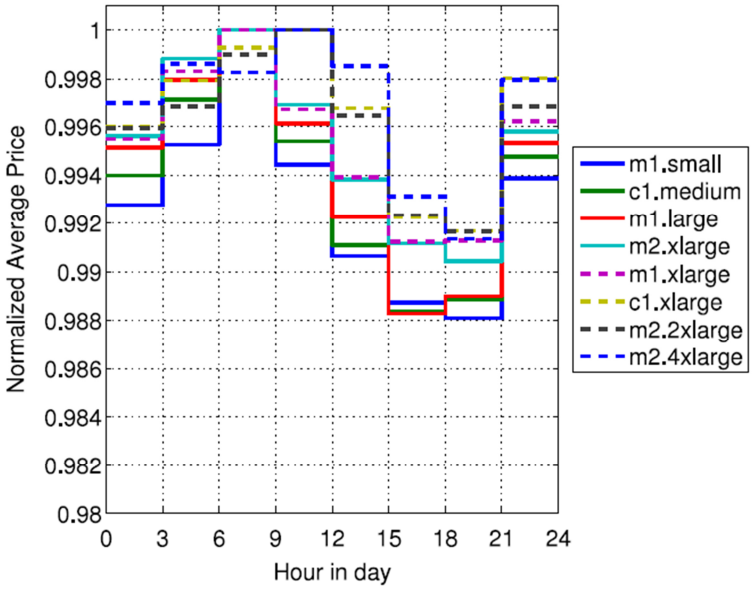


Figura 2: Variación de los precios a lo largo del día en EE.UU. Este en febrero de 2012 [B1].

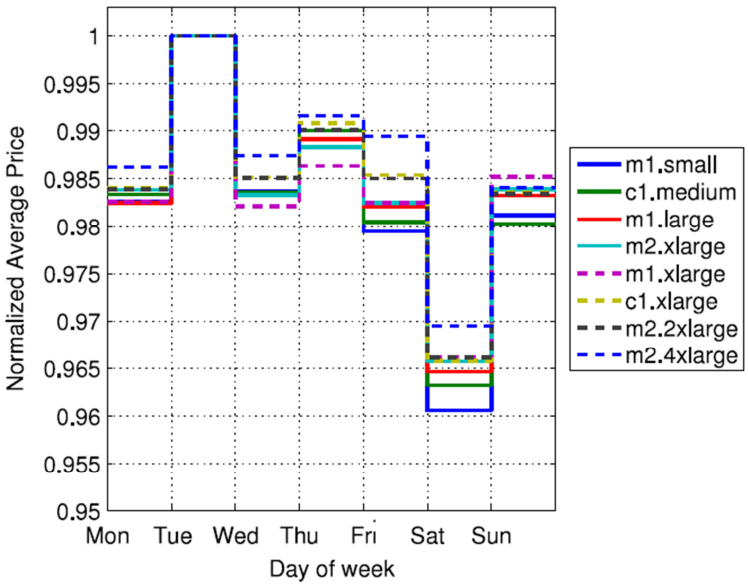


Figura 3: Variación de los precios a lo largo de la semana en EE.UU. Este en febrero de 2012 [B1].

2.1.5. *Checkpointing*

El *checkpointing* es una práctica en la cual guardamos el estado de un sistema (punto de control) con la posibilidad de volver a ese mismo estado en caso de error o fallo del sistema. Para el caso que nos aplica [B9] resulta interesante dicha práctica ante la posibilidad de perder los recursos que nos han sido asignados así como todo el cómputo realizado hasta el momento.

La utilidad del *checkpointing* en el contexto de las instancias puntuales se debe a que nos permite evitar malgastar tiempo de computación y dinero cuando una instancia es cancelada a mitad de la computación. Por tanto, este mecanismo resulta más interesante en tareas de larga duración donde las pérdidas serían mayores. Sin embargo, establecer una política de *checkpoints* en una tarea podría suponer un problema si no se puede definir un estado que salvaguardar cada X tiempo. Implementar un sistema de este tipo podría ser demasiado costoso o complejo y podría no ser rentable. Por esta razón, en la práctica, muy pocas aplicaciones permiten la utilización de *checkpoints* y un fallo en la instancia implica que la computación tiene que ser reiniciada desde el principio.

Una opción intermedia sería definir subtareas, la salida de la primera correspondería con la entrada de información que debe recibir la siguiente. De este modo se reducen las pérdidas en caso de pérdida de recursos sin implementar un sistema demasiado complejo.

Cuanto más corta sea una tarea menos probabilidades tendremos de no acabarla. Por tanto, la validez del uso de instancias puntuales encaja con la filosofía de los flujos de trabajo (*workflows*), subdivisión de una gran tarea en apartados más pequeños e independientes.

Este escenario de tareas cortas encaja completamente con el uso de las instancias puntuales, recursos que en periodos cortos de tiempo pueden mantenerse con un mayor porcentaje de éxito.

2.1.6. La puja de los 1000$

Ya se explicó en un apartado anterior cómo funciona la puja sobre las instancias. Las preguntas que se plantean ahora son:

1. ¿Deberían las pujas más altas aportar mayor disponibilidad?
2. ¿Podrían las pujas más altas modificar el comportamiento de los precios de las instancias?

En teoría, si un instancia cuesta 0,20€ y establecemos una puja de 1€ conseguiríamos más disponibilidad que si la puja fuese de 0,25€ ya que se cuenta con un margen

superior. Además, gracias a que Amazon cobra el precio de la instancia y no el precio de puja, el coste sería el mismo en ambos casos. Por otro lado, el hacer un número de peticiones considerable a un tipo de instancia hará que aumente el precio y podría dejar de ser la mejor opción.

En este respecto, es interesante comentar el caso de la "puja de los 1000$" [W11] para ilustrar los riesgos de establecer un precio de puja demasiado elevado. A priori podría parecer una puja lógica, las máquinas valen 0,20$ (por ejemplo) y con una puja infinitamente superior no la perderías nunca y sólo estarían cobrando 0,20$ la hora, mucho menos que una instancia bajo demanda. El problema de esta puja es que Amazon subió el precio de la instancia hasta los 1000$ y cobró 1 hora a dicho precio. Por tanto, si bien esta política de puja puede proporcionar una gran disponibilidad también puede originar un elevado gasto económico.

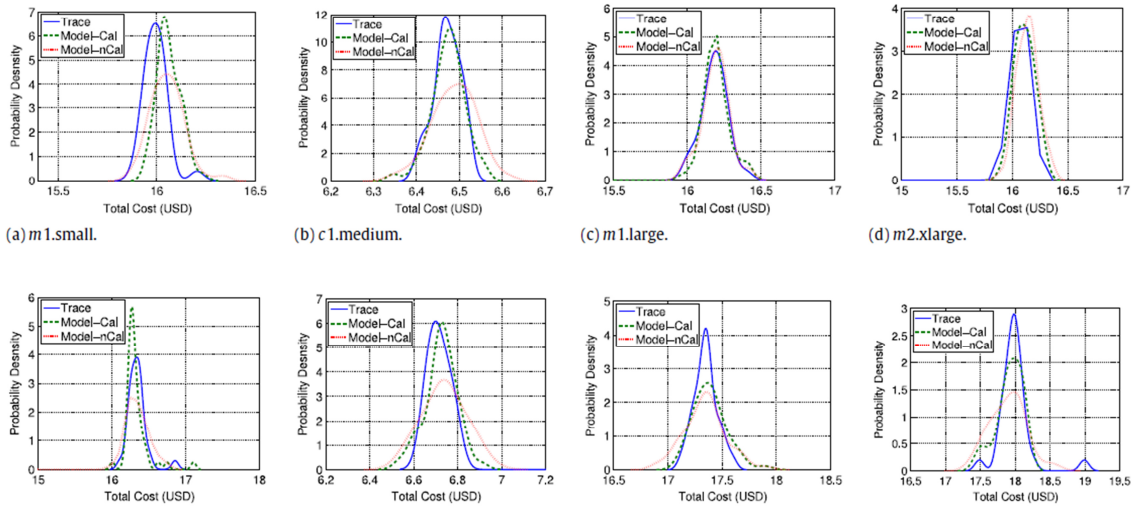
Este caso es una prueba más de la extraña naturaleza del funcionamiento interno de la variación de precios que hace Amazon y lo imprevisible que puede ser. Podemos concluir que pujar demasiado alto puede ser contraproducente a la hora de obtener disponibilidad a buen coste. Por tanto, es importante definir un precio de puja que nos asegure una buena disponibilidad sin incurrir en un coste demasiado elevado

2.1.7. Modelos matemáticos

En varias ocasiones se ha intentado establecer un modelo matemático con el que poder realizar una predicción del precio y poder ajustar al máximo las pujas sin renunciar a un grado alto de fiabilidad (mantenerse dentro de la puja) durante el máximo tiempo posible.

Se han seguido varias vías de enfoque, desde la estimación basada en históricos hasta poniéndose en el lado del proveedor (Amazon) e intentar razonar cómo harían un sistema similar al existente.

En ninguno de los casos se llega a una solución óptima para todos los posibles escenarios. En la siguiente figura se muestra cómo suelen funcionar dichos modelos en función del tipo de instancia.



(a) *m*1.small. (b) *c*1.medium. (c) *m*1.large. (d) *m*2.xlarge.

Figura 4: Resultados de pruebas de un modelo matemático [B6]

Cuanto más se ajusten las líneas verdes y rojas a la azul mejor es el modelo.

2.1.8. Conclusiones

Mediante el análisis anterior se llegan a una serie de conclusiones que se listan a continuación:

1. Las instancias puntuales son más económicas que las instancias bajo demanda.
2. Las ventajas económicas de las instancias puntuales tienen por contra un posible extra de tiempo necesario para ejecutar una misma tarea.
3. Las instancias puntuales son válidas para tareas de corta duración.
4. Incorporar un sistema de *checkpoints* puede resultar interesante pero también costoso. Como opción intermedia se pueden dividir las tareas en otras más pequeñas
5. Las horas no completadas no se cobran debido a que la máquina es desasignada
6. Una puja demasiado alta puede variar el precio aumentándolo.
7. Un aumento de la demanda de un tipo de máquina puede aumentar el precio.

El análisis anterior nos ha permitido extraer diferentes conclusiones que permiten definir los requisitos del sistema a desarrollar. Partiendo de estos puntos, se concluyó que el sistema a realizar debe ir enfocado a la realización de tareas de corta duración.

No se consideran los *checkpoints* algo imprescindible al tratarse de periodos cortos de tiempo y debido a que en la práctica las aplicaciones no suelen implementar este tipo de mecanismos. Por tanto, para dotar de mayor generalidad a la solución, se plantea un escenario en el que si una tarea es interrumpida a mitad, la misma debe ser reejecutada completamente. Por otro lado si se desean realizar tareas de larga

duración ya existen otro tipo de instancias (instancias reservadas) más adecuadas para ello.

Las variaciones de precio basadas en un aumento de la demanda se ignoran ya que el estudio sólo se hará con una instancia de forma simultánea.

Por último, consideramos que el cálculo de la viabilidad de una tarea no se puede basar en la definición de un modelo matemático, como se explicó anteriormente. Los motivos que nos llevan a tomar esta decisión son:

1. Ninguno de los modelos sirve para todos los entornos posibles.
2. No llegan a conocer 100% cómo funciona internamente el sistema de Amazon.
3. El sistema de Amazon podría sufrir un cambio que invalidase el modelo que esté usando y además sería sin previo aviso y consciencia de lo sucedido.

Basándonos en lo anterior y tras analizar diferentes estrategias de puja, concluimos que no hay ninguna que sea perfecta y que se adapte a los requisitos establecidos que estime si una tarea es posible dentro de un intervalo de tiempo y un presupuesto. Este sistema tendrá que ser de carácter general para no depender del funcionamiento interno (y desconocido) de Amazon y para evitar que se vea afectado ante posibles cambios. Para ello sus decisiones se basarán en función de los históricos de precios de los 2 últimos meses.

2.2. Visión general del sistema

En este apartado se explicará de forma general cual es el planteamiento de la aplicación o aplicaciones requeridas en el proyecto y posteriormente se hará un análisis específico de cada apartado donde se tratarán las decisiones de diseño.

En la figura 5 se muestra el sistema con 4 partes principales diferenciadas.

1. Una aplicación auxiliar que servirá para recoger los históricos de precios de los que dispone Amazon.
2. Un almacén de datos que de persistencia a los históricos de precios.
3. Un analizador de viabilidad de tareas.
4. Una aplicación auxiliar de simulación de tareas

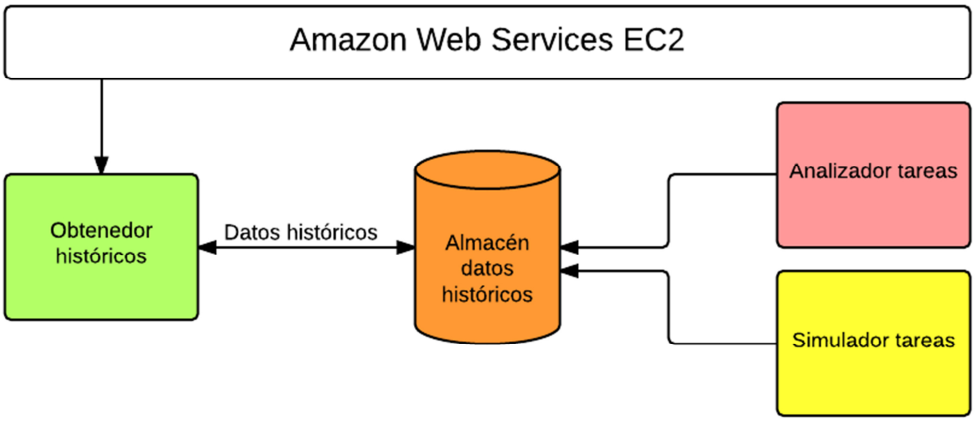


Figura 5: Esquema general del sistema.

Obtenedor de históricos: Esta aplicación auxiliar se encarga de solicitar los históricos de precios de las instancias puntuales, para cada zona, a los servicios web de Amazon y transmitirlos a un almacén de datos donde tengan persistencia. Su desarrollo fue necesario ya que era el único modo de manejar el volumen de datos requerido.

Analizador de tareas: El analizador de tareas es el encargado de dar información al usuario sobre una tarea y su viabilidad en un formato legible para ayudarle en la toma de decisiones. Toda la información que maneja está contenida en el almacén de datos históricos.

Simulador de tareas: Nos permite realizar multitud de pruebas sin coste alguno para todo tipo de escenarios. La simulación se realiza sobre los datos del almacén de datos históricos.

2.3. Requisitos del sistema

A continuación, establecemos los requisitos que debe cumplir el sistema a desarrollar.

Requisitos funcionales:

- RF1: El sistema debe ser capaz de obtener el histórico de precios de los servicios web de Amazon.

- RF2: Los históricos de precios deben ser persistentes.

- RF3: El sistema debe ser capaz de valorar si una tarea es posible dados unos parámetros determinados de tiempo y dinero.

- RF4: El sistema informará al usuario de la valoración que ha realizado de la tarea y solicitará confirmación para continuar, también dará la opción de

rechazar dicha ejecución.

- RF5: El sistema creará un pequeño informe de resultados informando de cómo ha ido la ejecución.

- RF6: Los históricos de precios deben estar actualizados hasta al día anterior al de la ejecución del sistema.

- RF7: Las aplicaciones que requieran como información de entrada la definición de una tarea deberán recibir mínimo la siguiente información:

    - Región y zona
    - Tipo de instancia
    - Sistema de la instancia
    - Nº de horas de la tarea
    - Tiempo disponible para finalizarla o fecha límite (siempre mayor o igual que el número de horas de la tarea).
    - Dinero total dispuesto a invertir en la tarea.

Requisitos no funcionales:

- RNF1: El sistema presentará la información sobre la viabilidad de las tareas en forma de gráfica.

- RNF2: Los históricos de precios deben actualizarse de forma autónoma y automática.

3. Diseño de la solución

En este apartado se describe el diseño de cada uno de los componentes que integran el sistema y la interacción que hay entre ellos.

3.1. Diseño general

En la figura 6 se muestra un diseño de los principales componentes que compondrán el sistema y cómo interactúan entre ellos.

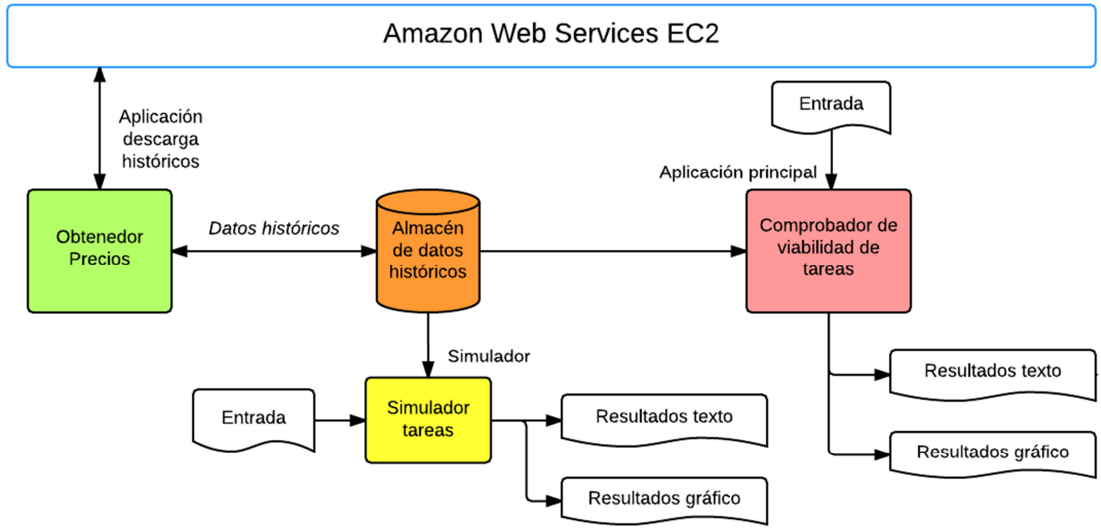


Figura 6: Esquema general del diseño

El diseño del sistema consta de 3 apartados bien diferenciados ya introducidos en el punto 2.2. En este nuevo esquema se aprecia que el comprobador de viabilidad y el simulador reciben una información de entrada desde el exterior y generan unos resultados en formato de texto y gráfico.

La información que se define en ambos casos en la entrada es la especificación de una tarea a ejecutar. Dicha información deberá cumplir el requisito funcional número 7 del apartado 2.3. El precio de la puja se calcula en función de las horas de la tarea y el dinero total disponible.

La información de salida tiene como función ayudar al usuario en la toma de decisiones. En el caso de la aplicación principal esta información de salida tiene como función mostrar al usuario una valoración sobre si es buena idea o no la ejecución de una tarea en unas determinadas circunstancias. Para el caso del simulador la información de salida sirve para valorar los cálculos y resultados que arroja la aplicación principal.

3.2. Cálculo de posibilidades de tareas

Como se mencionó en apartados anteriores, los modelos matemáticos que se habían diseñado para la previsión de los precios de las instancias no eran válidos para todas las instancias posibles por lo que se veían limitados en este aspecto. Además quedaban totalmente invalidados ante un cambio repentino en el funcionamiento interno de Amazon.

Por estos motivos se creó un mecanismo de estimación de viabilidad propio el cual está basado en el histórico de precios al que Amazon nos da acceso. La primera pregunta que surge en estos momentos es: "¿Hasta cuándo debe remontarse el histórico?". ¿Sería necesario considerar los 3 últimos años de actividad de las instancias para garantizar un buen cálculo de posibilidades? La respuesta es no. Del mismo modo que se pretende establecer un sistema de cálculo general que esté preparado ante cambios en los algoritmos de Amazon no podemos considerar una cantidad de datos tan elevado y distante en el tiempo por los mismos motivos. No tenemos certeza de cómo funcionaba dicho sistema hace 1 o 2 años y podrían afectar negativamente al cálculo final. Por lo tanto se consideran los 2 últimos meses (datos siempre disponibles mediante los servicios web de Amazon) suficientes para cualquier estimación oportuna.

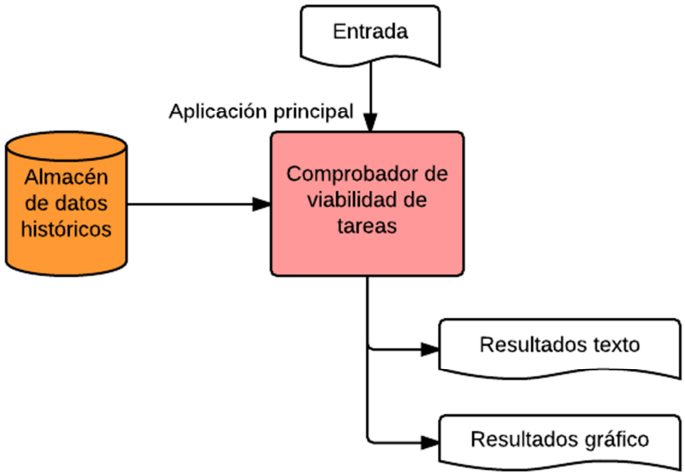


Figura 7: Esquema diseño aplicación principal

En el diseño que aparece en la figura 7 se aprecia que el histórico de precios se guarda en un almacén de datos, ¿es realmente necesario? Podríamos usar la propia aplicación encargada de obtener los precios dentro de la aplicación principal y tener siempre los últimos datos actualizados. El problema es que la descarga de estos históricos tarda entre unas 3 y 4 horas por lo que lo hace totalmente inviable. Por el contrario, con el uso de la base de datos se pueden realizar consultas de miles de datos en escasos segundos además de tenerlas en caché una vez realizadas. Este último punto es realmente útil en los casos en los que es necesario realizar más de una consulta ya que pasan a ser casi instantáneas.

Una vez quedan decididos los datos que se van a utilizar para las estimaciones queda definir cómo se van a realizar y cómo se va a valorar la viabilidad de ejecutar una tarea en cada una de las zonas de una región. Para la valoración de las tareas se creó un sistema de puntuaciones, a mayor puntuación mayores posibilidades de completar la tarea dentro de los parámetros establecidos en la entrada por el usuario.

En el siguiente ejemplo se explica cómo funciona. Supongamos que queremos gastar como máximo 1€ en realizar una tarea de 4 horas de duración y que tenemos un tiempo máximo de 10 horas para completar la misma. Para ello vamos a utilizar las instancias puntuales.

En primer lugar, se fija el precio de puja de acuerdo al presupuesto y la duración de la tarea. En este caso, la aplicación fijaría el precio de puja en 25 céntimos (1€ / 4 horas).

Con la puja ya fijada, se mira el histórico de precios almacenado en la base de datos. El algoritmo busca espacios de tiempo de la misma duración que la tarea (4 horas en este caso) donde el precio de puja supere siempre al precio de la instancia puntual. Cada vez que encuentra un intervalo donde se dan estas condiciones añade 1 a la puntuación. Es decir, mira si en los dos últimos meses se han dado condiciones similares y la frecuencia con la que sucedieron. Una puntuación de 5 indicaría en este caso que encontró 5 intervalos de 4 horas donde el precio de la instancia era menor a 0,50€. Por simplicidad y coherencia con el sistema de cobro de Amazon, al calcular la puntuación se miran tan sólo horas completas.

Ejemplo: En la figura 8 se muestran gráficamente todos los intervalos en los que se podría completar la tarea con éxito.

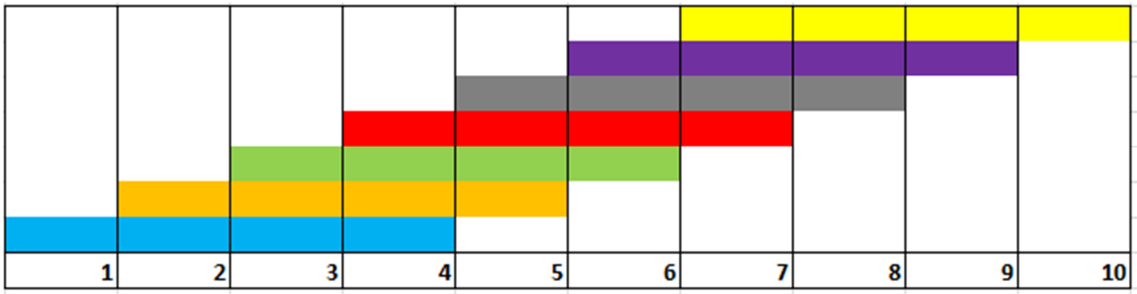


Figura 8: Posibles intervalos de ejecución

En este ejemplo vemos 7 intervalos posibles (debido a los parámetros de duración de tarea y tiempo límite establecidos), por lo tanto la puntuación máxima sería 7. Se comprueba en cada intervalo si la ejecución de la tarea es posible. Si es posible en 5 de esos 7 intervalos la puntuación es 5/7.

Como apuntamos anteriormente, se toman los 2 últimos meses de los históricos para hacer la valoración de la tarea, que equivalen a 8 semanas. Sin embargo, no se consideran todos los datos de los últimos meses. Sólo se consideran los datos correspondientes al mismo momento temporal con respecto al instante actual. De esta forma, intentamos que el cálculo de la viabilidad tenga en cuenta situaciones que se acerquen lo máximo posible a la situación actual. La decisión de realizar el cálculo teniendo en cuenta el mismo momento temporal pero en las 8 semanas anteriores se debe a la existencia de comportamientos periódicos y repetitivos semanalmente [B6].

A raíz de hacer un cálculo por semana entran en juego nuevas consideraciones a tener en cuenta para realizar el cálculo de la viabilidad de forma correcta que se listan a continuación:

1. Los históricos de la base de datos empiezan en jueves pero hoy es lunes, por tanto dichos intervalos calculados no se corresponden a una situación similar.

   En la figura 9 se puede ver de forma gráfica la explicación de este punto. En este caso observamos una tarea con un tiempo límite de 2 días. El intervalo verde debería ser el punto inicial de observación de los históricos. El intervalo rojo comienza en el primer dato que contienen los históricos que en este caso no es correcto.

Figura 9: Fechas de inicio correctas e incorrectas en la consulta de históricos

2. Los cálculos empiezan usando históricos con fechas similares (mismo día de la

semana) pero una vez pasado el tiempo límite se vuelve a empezar en fechas que no corresponden. Ejemplo: Empezamos en miércoles y tenemos 2 días para la tarea, tras el viernes habría que volver a situarse en el siguiente miércoles.

El total de intervalos que se deben observar en los históricos queda representado en la figura 10 que se muestra a continuación.

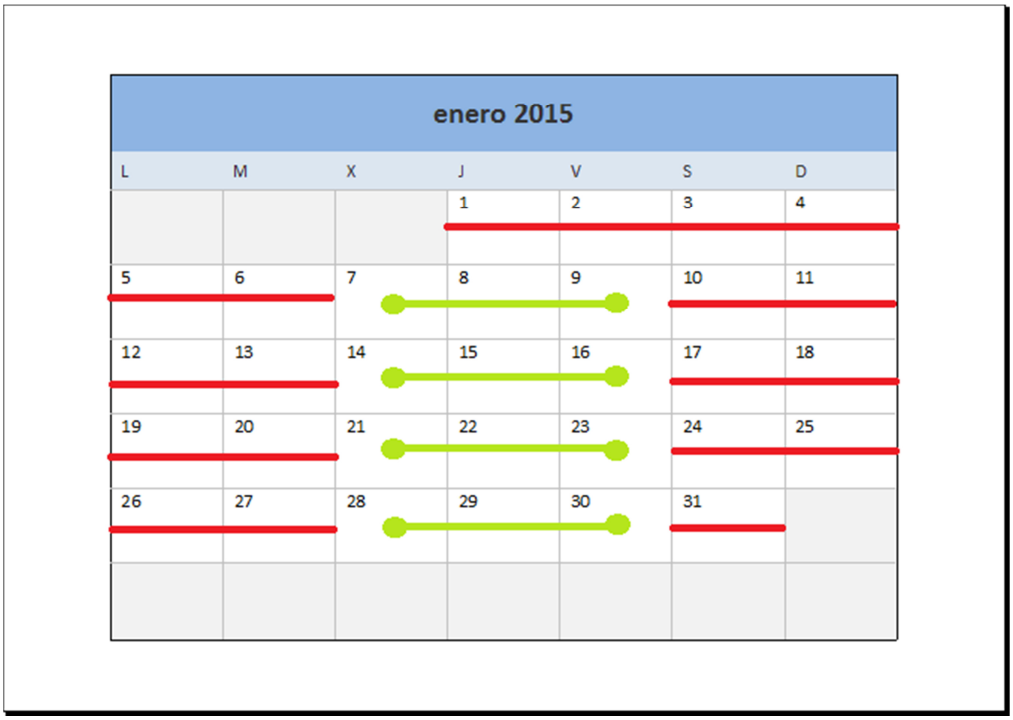


Figura 10: Total de intervalos a revisar.

3. Los cálculos parten de un miércoles y se tienen 8 días de tiempo límite. Al acabar de realizar cálculos desde el primer miércoles estaríamos en jueves. Partiendo del punto anterior habría que situarse en el próximo miércoles para seguir utilizando datos similares a los que se va a encontrar la aplicación una vez empiece su ejecución. El problema en este caso es que se está ignorando un miércoles y por tanto la salida que dé al usuario será menos veraz y con una puntuación peor a la real.

   En la figura 11 se observa que aunque se hayan recorrido los históricos de varios miércoles en este caso en un solo intervalo (de 8 días en el ejemplo) no hay que ignorar ningún comienzo posible similar al día de la semana en el que se realiza la ejecución.

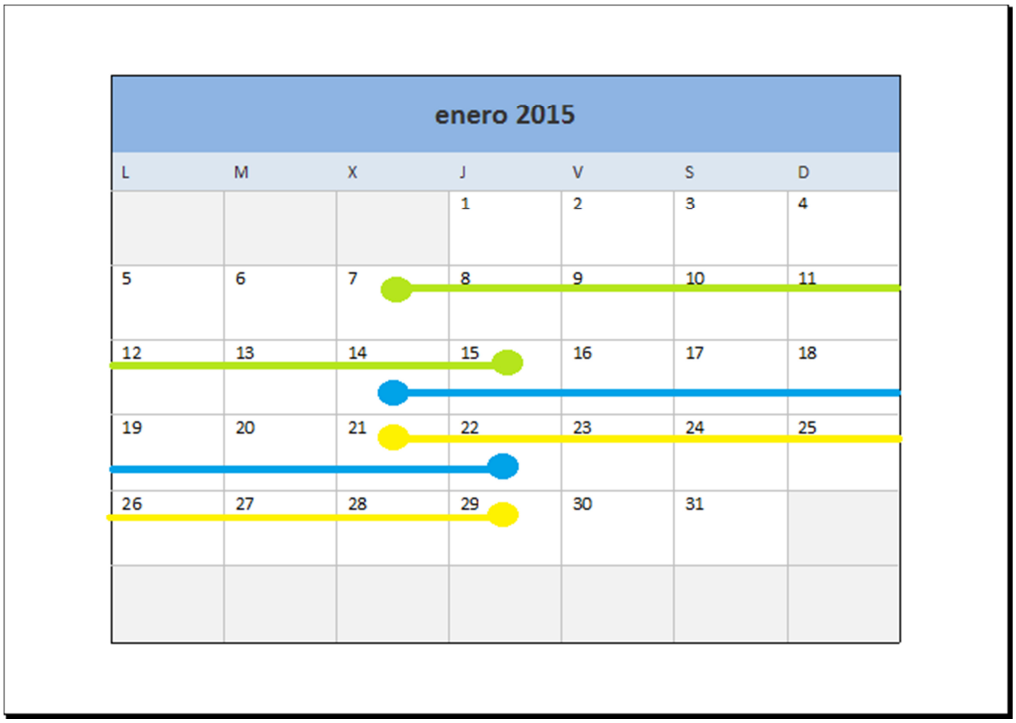


Figura 11: Intervalos mayores de una semana.

Como los históricos contienen más de un lunes, martes, y del resto de días de la semana, el algoritmo permite situarse en el segundo lunes (por ejemplo) dado que también representaría una situación similar.

Por último queda definir cómo se calcula la valoración final. Se consideró que no podían tener el mismo peso lo datos de la semana más lejana temporalmente que los datos más recientes. Para ello se fijaron unos pesos por los cuales los datos más recientes tendrían más repercusión en el resultado final, se listan a continuación en la tabla 2:

Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 6	Semana 7	Semana 8
50%	25%	12%	6%	3%	2%	1%	1%

Tabla 2: Distribución de pesos por semanas.

La semana 1 corresponde con la más reciente mientras que la semana 8 con la menos reciente. El total de porcentajes suma 100%.

4. Implementación de la solución

En esta sección se explica de forma general cómo se ha implementado cada componente. Se habla sobre qué tecnologías se han usado además de cómo está organizado el contenido del almacén de datos.

Puede encontrarse más información en el anexo I de la memoria.

En la figura 12 se puede ver el esquema general de implementación. El sistema está compuesto por 3 componentes principales que son:

- Aplicación principal (predictor).
- Aplicación de obtención de históricos de precios.
- Simulador de tareas.

Los 3 componentes interactúan con una base de datos común donde obtienen la información necesaria para su ejecución.

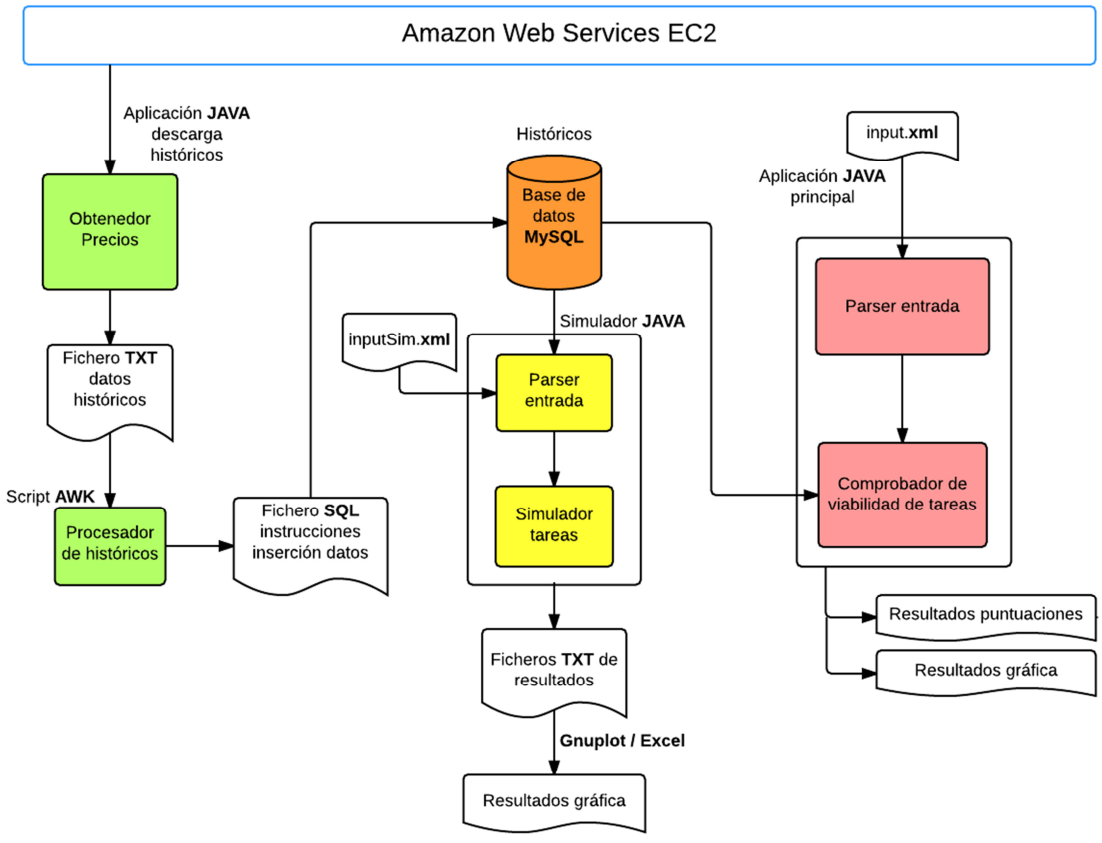


Figura 12: Implementación de la solución.

Para la aplicación principal se optó por una solución basada en Java por el rápido acceso al tratarse de un lenguaje de sobra conocido. Además Amazon tiene un SDK con

instrucciones sencillas para su instalación en Eclipse [W3] (entorno de desarrollo utilizado en este caso) que facilita el uso de sus servicios web.

La aplicación del histórico no es la encargada de procesar los datos descargados como se aprecia en el esquema. Para este cometido se optó por crear unos scripts en AWK que se encargan de transformar dichos históricos en instrucciones SQL de inserción para la base de datos. El propio procesado de los históricos se podría haberlo realizado en la aplicación Java pero se consideró preferible el uso de scripts con vistas a posibles cambios futuros y sobre todo por temas de rendimiento general, estamos hablando de más de 1 millón de datos que procesar si consideramos un log de 2 meses. En el caso de tener que modificar el procedimiento de generación de un fichero SQL que contenga las instrucciones de inserción de datos no se tendría que modificar para nada la aplicación Java y únicamente se tendría que ajustar un script. De este modo el mantenimiento resulta más sencillo y modular.

Para el simulador se optó por una solución similar a la de la aplicación principal, una aplicación JAVA cuya entrada viene dada en un fichero XML. Los resultados son generados en un fichero de texto al esperar un volumen de datos más grande que en la aplicación principal en el caso de realizar numerosas pruebas. Además este fichero de texto nos facilita poder trasladar esos datos a gráficas posteriormente del modo que creamos oportuno.

Por último tenemos la base de datos. Se eligió MySQL por simplicidad, ya estaba familiarizado con MySQL y no se requería de nada realmente complejo en este apartado.

En los siguientes apartados detallamos los aspectos más importantes de implementación de cada uno de los componentes anteriores.

4.1 Aplicación principal

La aplicación que realiza la previsión de viabilidad de cada zona requiere de un fichero XML de entrada donde se fijan los parámetros a ejecutar. A continuación, la tabla 3 muestra la estructura del fichero XML.

```
<?xml version="1.0"?>
<input>
        <launcher>ruta script lanzador</launcher>
        <ami>imagen a usar</ami>
        <si_type>r3_4xlarge</si_type>
        <system>Linux/UNIX (Amazon VPC)</system>
```

```
        <cost>1.25</cost>
        <deadline_year>2015</deadline_year>
        <deadline_month>11</deadline_month>
        <deadline_day>05</deadline_day>
        <deadline_hour>9</deadline_hour>
        <deadline_minute>00</deadline_minute>
        <deadline_second>00</deadline_second>
        <task_time>5</task_time>
        <zone>AP</zone>
</input>
```

Tabla 3: Diseño obtenedor de precios

Es necesario que la base de datos esté actualizada hasta la semana anterior a la del día en el que se lanza la ejecución para su correcto funcionamiento. Este punto es importante ya que los cálculos los realiza sobre las últimas 8 semanas de históricos (las más recientes).

Una vez realizados los cálculos (con la lógica descrita en el punto 4.2 de la memoria) proporciona dos salidas, una de texto y otra en forma de gráfica que resulta más visual e inmediata. En las siguientes figuras puede verse un ejemplo de las salidas.

```
RESULTS
------------------------------
*** ap-northeast-1a ***
 Week: 1/8; Score: 29/151 -> 19,21%
 Week: 2/8; Score: 49/151 -> 32,45%
 Week: 3/8; Score: 83/151 -> 54,97%
 Week: 4/8; Score: 63/151 -> 41,72%
 Week: 5/8; Score: 87/151 -> 57,62%
 Week: 6/8; Score: 140/151 -> 92,72%
 Week: 7/8; Score: 80/151 -> 52,98%
 Week: 8/8; Score: 25/151 -> 16,56%

*** ap-northeast-1c ***
 Week: 1/8; Score: 94/151 -> 62,25%
 Week: 2/8; Score: 0/151 -> 0,00%
 Week: 3/8; Score: 69/151 -> 45,70%
 Week: 4/8; Score: 142/151 -> 94,04%
 Week: 5/8; Score: 116/151 -> 76,82%
 Week: 6/8; Score: 61/151 -> 40,40%
 Week: 7/8; Score: 25/151 -> 16,56%
 Week: 8/8; Score: 145/151 -> 96,03%

*** ap-northeast-1b ***
 Week: 1/3; Score: 0/151 -> 0,00%
 Week: 2/3; Score: 0/151 -> 0,00%
 Week: 3/3; Score: 0/151 -> 0,00%

Week 1:                    ap-northeast-1a: 29.0 | ap-northeast-1c: 94.0 | ap-northeast-1b: 0.0 |
Week 4:                    ap-northeast-1a: 56.0 | ap-northeast-1c: 76.0 | ap-northeast-1b: 0.0 |
All weeks:                 ap-northeast-1a: 69.0 | ap-northeast-1c: 81.0 | ap-northeast-1b: 0.0 |

Final SCORES:              ap-northeast-1a: 58.8 |   ap-northeast-1c: 99.6|    ap-northeast-1b: 0.0 |
```

Figura 13: Salida de texto del predictor

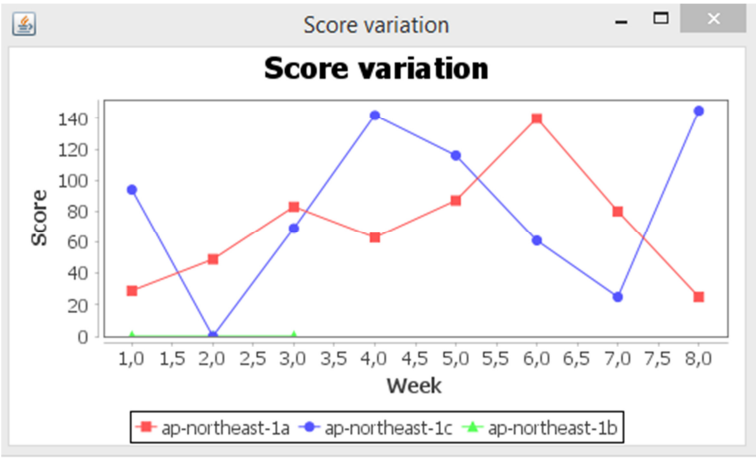


Figura 14: Salida gráfica del predictor

Para cada semana se muestra la puntuación de forma individual. En los resultados finales "Week 1" indica la puntuación tomando sólo una semana, "Week 4" indica que es la ponderación de las 4 primeras semanas y "All weeks" que toma en cuenta todas ellas. Este desglose es incluido para no tener únicamente la puntuación final y poder valorarlo en su totalidad.

En este caso en particular se ve que en una de las zonas no hay información para todas las semanas, por lo tanto se representa únicamente la información de la que se dispone.

4.2 Obtenedor de precios

El obtenedor está directamente ligado con la base de datos y necesita de esta para poder operar. A la hora de descargar los históricos mira la tabla de actualizaciones de la base de datos (descrita en el apartado 4.4, figura 18) y fija dicha fecha como límite al solicitar los históricos. En la figura 15 se muestra el diseño de este componente.

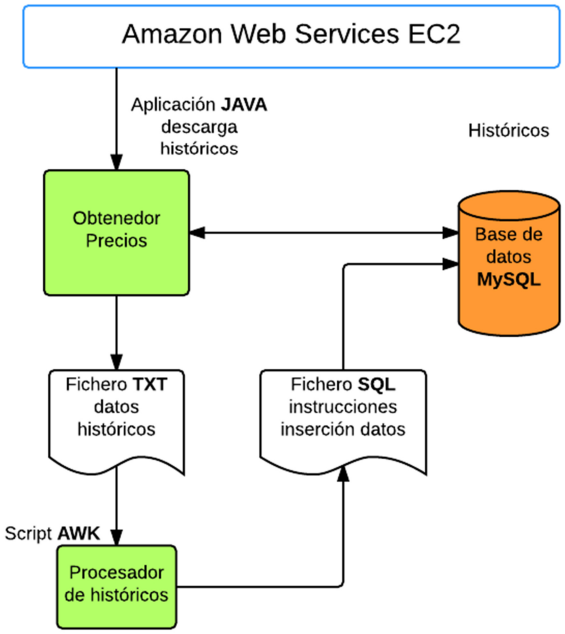


Figura 15: Diseño obtenedor de precios

La salida del programa es un fichero de texto para cada una de las zonas. A continuación se muestra cómo son los históricos de precios de las instancias puntuales que pueden obtenerse.

En la tabla 4 se muestra una línea de los históricos descargados mediante la aplicación de obtención tal y como es recibida.

{InstanceType:r3.8xlarge,ProductDescription:Linux/UNIX,SpotPrice:0.512500,Timestamp: Fri Oct 16 21:00:06 CEST 2015,AvailabilityZone: ap-northeast-1a}

Tabla 4: Línea del histórico de precios

Para introducir los históricos en la base de datos primero se procesan y se transforman en un fichero SQL con instrucciones de inserción. El procesado se realizó mediante scripts en AWK, puede encontrarse más información al respecto en el anexo I de la memoria.

4.3 Simulador de tareas

El simulador, al igual que el predictor, toma la información de entrada de un fichero XML. A continuación, la Tabla 5, muestra su contenido.

```
<?xml version="1.0"?>
<input>
        <si_type>c1_medium_US</si_type>
        <system>Linux/UNIX</system>
        <cost>10 </cost>
        <init_year>2014</init_year>
        <init_month>6</init_month>
        <init_day>30</init_day>
        <init_hour>10</init_hour>
        <init_minute>00</init_minute>
        <init_second>00</init_second>
        <deadline>24</deadline>
        <task_time>5</task_time>
        <subzone>us-west-2c</subzone>
</input>
```

Tabla 5: Estructura del fichero XML de entrada

El simulador fija la puja dividiendo el dinero disponible entre las horas que queremos mantener la instancia. Cuando una instancia es asignada la tarea se da por comenzada

y no vuelve a relanzarse, es decir, si la instancia se mantiene el tiempo total de la tarea se considera éxito y si los recursos se pierden en algún momento se considera fracaso.

La salida del simulador es en formato de texto, informando sobre cómo fue la tarea. En la figura 16 puede verse la salida del simulador.

```
Task info:
---------------------------------
Spot Instance: c1_medium_AP
Cost (money): 0.21
Init date: 2015-07-01 01:00:00.217
Task time (hours): 10
Zone: ap-northeast-1c

PUJA: 0,0210

FECHA INICIO SOLICITADA: 2015-07-01 01:00:00.217
FECHA INICIO TAREA: 2015-07-01 01:00:00.217
FECHA FIN TAREA: 2015-07-01 11:00:00.217
RESULTADO;COSTE TOTAL: EXITO;0,2031
```

Figura 16: Salida del programa de simulación

Por otro lado, se incluyeron nuevas salidas para el simulador cuando se realizaron baterías de pruebas con miles de tareas ejecutadas. Los resultados se trasladaron a ficheros de texto con el fin de poder manipularlos más fácilmente. Puede leerse más sobre este aspecto en el anexo I de la memoria.

4.4 Base de datos

Para poder almacenar los datos de forma persistente y asegurar un acceso rápido a ellos se implementó una base de datos haciendo uso de MySQL, concretamente la versión 5.6

La base de datos cuenta con tablas para almacenar todos los precios de todas las máquinas de cada zona. El nombre de cada tabla es representativo ya que en él se incluye el tipo de máquina así como la región en la que se encuentra. Siguen la siguiente estructura:

tipo_tamaño_region

Podría almacenarse todo en una misma tabla ya que el contenido en todas ellas es el mismo pero aumentaría notablemente los datos que traemos de la base de datos notablemente.

Algunos ejemplos de nombres:

- C1_MEDIUM_US

- C1_MEDIUM_UE
- M1_XLARGE_SA
- R3_8XLARGE_AP

Hay que tener en cuenta que en cada una de las regiones existen varias zonas. Por ejemplo en los centros de datos de Estados Unidos podemos encontrar las zonas “us-west-2a” y “us-west-2b” entre otras, la información de ambas zonas se concentrará en una única tabla para cada tipo de máquina.
La información que se guarda en las tablas es la siguiente:

- *Zone* : zona específica en la que se encuentra la instancia.
- *Description*: describe el sistema que utiliza la instancia. Existen varios tipos
  - Linux/UNIX
  - SUSELinux
  - Windows
- *Price*: precio de la instancia.
- *Time*: fecha del registro.
- *WeekDay*: día de la semana correspondiente a la fecha del campo “Time”.
- *TimeZone*: zona horaria (CET, CEST, etc…).

A continuación, la tabla 6 muestra un código SQL de ejemplo para la creación de una de las tablas que almacenan los históricos de las instancias:

```
CREATE TABLE IF NOT EXISTS c1_medium_AP
(
        id              BIGINT NOT NULL AUTO_INCREMENT,
        Zone            VARCHAR(20) NOT NULL,
        Description     VARCHAR(40) NOT NULL,
        Price           FLOAT NOT NULL,
        Time            TIMESTAMP NOT NULL,
        WeekDay         VARCHAR(5) NOT NULL,
        TimeZone        VARCHAR(5) NOT NULL,
        PRIMARY KEY (id)
);
```

Tabla 6: Ejemplo de código SQL de creación de tabla de instancia

Puede observarse que aparece un nuevo campo “id”. Este campo sirve para diferenciar cada uno de los registros. Se incorporó para evitar usar una clave compuesta por los campos “Zone”, “Description” y “Time” que resultaba más ineficiente sin aportar nada realmente útil.

En la siguiente figura puede verse el contenido de una de las tablas de históricos:

id	Zone	Description	Price	Time	WeekDay	TimeZone
54865	us-west-2b	Linux/UNIX	0.2563	2014-06-30 10:00:47	Mon	CEST
54857	us-west-2a	Linux/UNIX	0.2579	2014-06-30 10:26:48	Mon	CEST
54837	us-west-2a	Linux/UNIX	0.2568	2014-06-30 10:33:10	Mon	CEST
54838	us-west-2c	Linux/UNIX	0.2573	2014-06-30 10:33:10	Mon	CEST
54823	us-west-2a	Linux/UNIX	0.2578	2014-06-30 10:37:23	Mon	CEST
54819	us-west-2c	Linux/UNIX	0.2574	2014-06-30 10:39:27	Mon	CEST
54817	us-west-2c	Linux/UNIX	0.2573	2014-06-30 10:41:30	Mon	CEST
54805	us-west-2a	Linux/UNIX	0.2569	2014-06-30 10:49:48	Mon	CEST

Figura 17: Contenido de las tablas de instancias

Además de las tablas que almacenan la información histórica de las instancias se creó una auxiliar para llevar un control de hasta qué fecha se había actualizado cada una de las zonas. En la tabla 7 se muestra el código SQL para su creación.

```
CREATE TABLE IF NOT EXISTS zone_updates (
        Zone            VARCHAR(20) NOT NULL,
        Time            TIMESTAMP NOT NULL,
        PRIMARY KEY (Zone)
);
```

Tabla 7: Ejemplo de código SQL de creación de tabla de instancia

A continuación, la figura 18 muestra un ejemplo del contenido de la tabla de actualizaciones:

Zone	Time
AP_NORTHEAST_1	2014-12-15 11:22:03
AP_SOUTHEAST_1	2014-12-15 11:22:49
AP_SOUTHEAST_2	2014-12-15 11:22:43
EU_CENTRAL_1	2014-12-15 11:08:22
EU_WEST_1	2014-12-15 10:57:21
SA_EAST_1	2014-12-15 15:49:35
US_EAST_1	2014-12-15 11:15:40
US_WEST_1	2014-12-15 11:17:21
US_WEST_2	2014-12-15 11:18:04

Figura 18: Tabla de actualizaciones

5. Verificación y evaluación del sistema

En este apartado de la memoria se detallan las pruebas realizadas y las conclusiones extraídas a partir de ellas.

5.1. Verificación del sistema

Para la verificación del sistema creado se van a realizar múltiples pruebas con el predictor y el simulador en diferentes escenarios y con diferentes tipos de instancias.

En primer lugar, se ha realizado una batería exhaustiva de pruebas utilizando el simulador para analizar la utilidad del predictor desarrollado.

Con el objetivo de explorar la eficacia de la solución propuesta de forma independientemente del instante temporal en el que se realiza, cada uno de los escenarios establecidos se simula considerando 350 instantes temporales distintos. Ese número de pruebas fue seleccionado para tener una confianza del 95% considerando que el espacio muestral son todas las posibles horas de los 2 meses utilizados como espacio de prueba [W13]. Por tanto, en cada una de las 350 ejecuciones se parte de un punto distinto. En un momento inicial se fija una fecha a partir de la cual se empieza a ejecutar la simulación de la tarea. Cuando esa prueba finaliza el tiempo inicial aumenta en 3h, es decir, si la primera prueba se hizo a las 00:00 del día 1 la segunda se hará a las 03:00 del día 1.

Con estas variaciones las pruebas abarcan 1 mes y medio de puntos iniciales distintos.
Para cada uno de los escenarios analizados, definidos a continuación, el tiempo de ejecución de la tarea así como el tiempo total para ejecutarla (*deadline*) es distinto. Resulta interesante comparar los resultados con diferente *deadline* al dar más margen de inicio en la ejecución de la tarea y no forzar a que sea inmediato, se espera averiguar en qué medida puede afectar. El tiempo inicial de la tarea viene definido en la entrada que recibe el simulador. Los tiempos de las tareas van desde 5 horas hasta 10. Consideramos esos tiempos ya que este tipo de instancias son adecuadas para la gestión de tareas de corta duración.

En un primer momento se fija un *deadline* igual al tiempo de la tarea (primer escenario). Posteriormente se fija un *deadline* mayor que el tiempo de la tarea (el doble del tiempo necesario para ejecutar la tarea).

La simulación se compara con los datos del predictor, que nos indica qué zona es mejor en cada caso. Para que resulte lo más real posible tomamos 4 meses de históricos y los dividimos entre el predictor y el simulador. Así pues, el predictor trabaja con los dos primeros meses y el simulador con los dos siguientes. De este

modo se pretende que el predictor no tenga en cuenta los datos “futuros” al analizar la tarea.

Además, tomando varios meses para evaluar las pujas del predictor podemos seleccionar diferentes instantes temporales y dotar a los resultados de una mayor fiabilidad ya que el predictor se evalúa bajo diferentes condiciones de utilización de los recursos *cloud* (diferentes precios de las instancias puntuales), en diferentes días y horas y en varias semanas.

Las fechas seleccionadas para el predictor son:

- Inicio: 06/07/2015
- Fin: 31/08/2015

Las fechas seleccionadas para el simulador son:

- Inicio: 01/09/2015
- Fin: 15/10/2015

Para la evaluación seleccionamos 3 máquinas de diferente tipo y variación de precios. Las máquinas elegidas son las siguientes:

- C1 medium
- C4 2xlarge
- M4 4xlarge

Consideramos estos 3 tipos de instancia por mostrar diferentes comportamientos en las variaciones de precio, de este modo contemplamos escenarios diferentes que pueden servir para otras instancias con comportamientos similares. Para las pruebas, se eligió la región de Asia Pacífico (Tokio), la elección de la región no resulta decisiva mientras las máquinas elegidas contemplen diferentes comportamientos.

Los precios de puja estimados, en cada caso, se extraen de unas gráficas generadas a partir de los históricos con Gnuplot y son una aproximación ligeramente superior del precio que más se repite a lo largo del tiempo.

En los siguientes apartados se detallan las pruebas realizadas para cada una de las instancias anteriores.

Pruebas con C1 medium

En la figura 19 se muestra el comportamiento en la variación de precios para la instancia c1.medium en la región de Asia Pacífico (Tokio). En el gráfico se aprecia que el precio apenas varía en ambos casos aunque una de ellas se mantiene más estable además de tener precios más bajos.

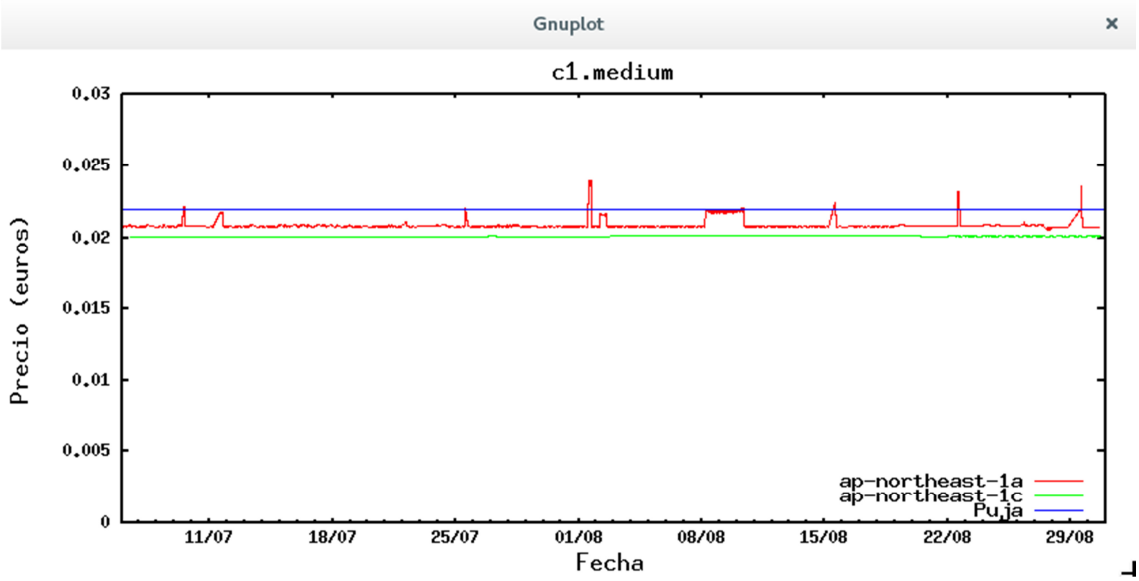


Figura 19: Histórico de precio de c1.medium durante las fechas seleccionadas para el predictor.

Precio instancia bajo demanda: 0.14€/h
Puja estimada: 0.022€

COSTE TOTAL ESTIMADO	5 horas	6 horas	7 horas	8 horas	9 horas	10 horas
Coste instancia puntual	0,11€	0,132€	0,154€	0,176€	0,198€	0,22€
Coste instancia bajo demanda	0,7€	0,84€	0,98€	1,12€	1,26€	1,4€

Tabla 8: Comparación de presupuesto entre instancias puntuales e instancias bajo demanda.

Resultados del predictor

En la figura 20 se muestran los resultados del predictor para la zona *ap-northeast-1a* en tareas desde 5 a 10 horas con ambos *deadlines*.

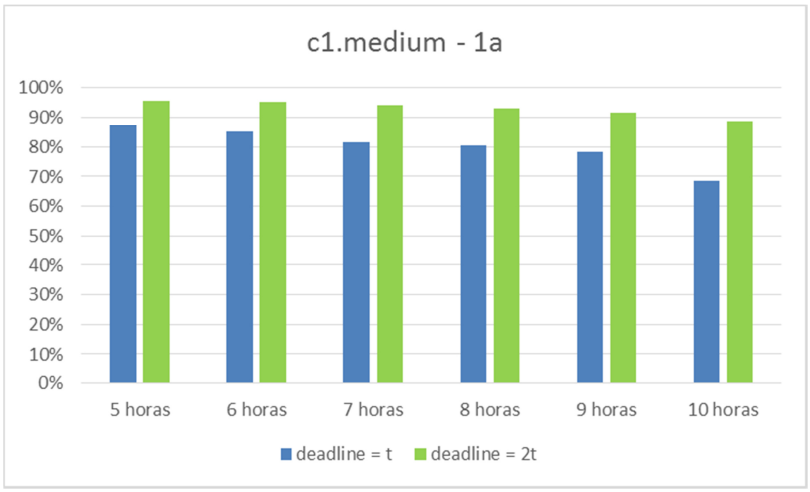


Figura 20: Resultados de la predicción en ap-northeast-1a con *deadline* = t y *deadline* = 2t

En la figura 21 se muestran los resultados del predictor para la zona *ap-northeast-1c* en tareas desde 5 a 10 horas con ambos *deadlines*.

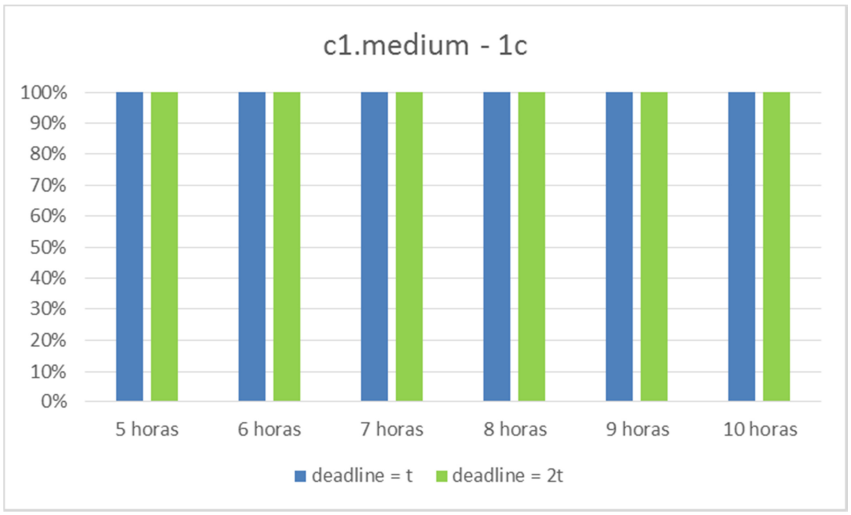


Figura 21: Resultados de la predicción en ap-northeast-1c con *deadline* = t y *deadline* = 2t

En este primer caso que nos ocupa no queda lugar a dudas, la zona *ap-northeast-1c es* la mejor con la máxima puntuación posible (figura 21). Aumentamos el *deadline* para comprobar en qué medida pueden mejorarse los resultados de la otra zona. Al aumentarlo, obtenemos un incremento en torno al 10%, dejando dicha zona con una puntuación realmente buena con respecto al anterior resultado. Por tanto, establecemos que un aumento del *deadline* aumenta las posibilidades de éxito de ejecución de una tarea.

Resultados de la simulación

En la figura 22 se muestran los porcentajes de éxito extraídos de la simulación para todas las duraciones de las tareas con *deadline* = t y *deadline* = 2t en ap-northeast-1a.

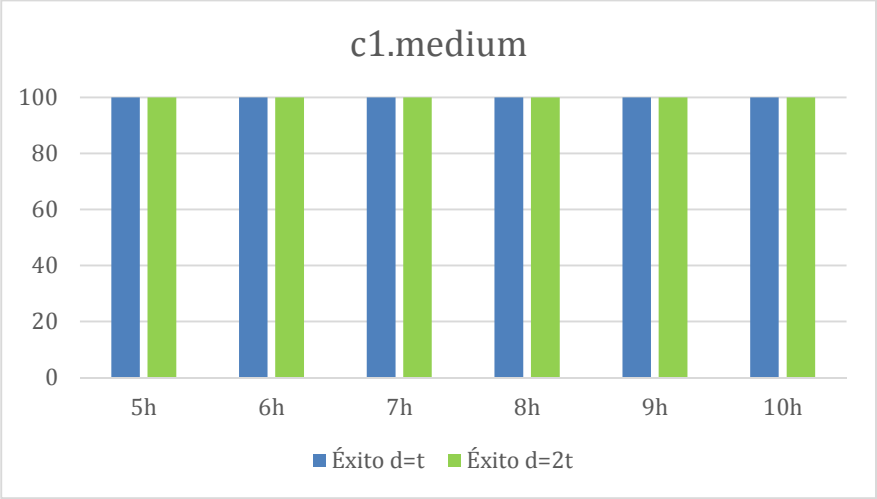


Figura 22: Porcentaje de éxito en la simulación con *deadline* = t y *deadline* = 2t en ap-northeast-1a

En la figura 23 se muestran los porcentajes de éxito extraídos de la simulación para todas las duraciones de las tareas con *deadline* = t y *deadline* = 2t en ap-northeast-1c.

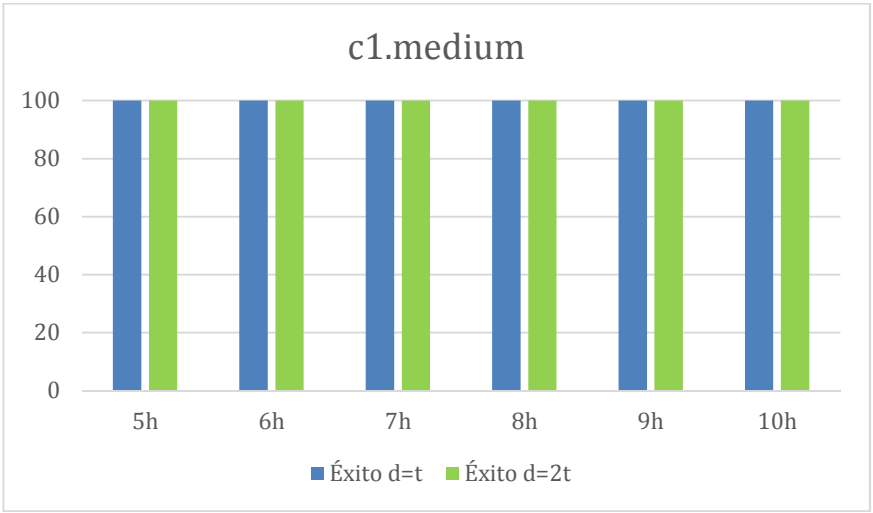


Figura 23: Porcentaje de éxito en la simulación con *deadline* = t y *deadline* = 2t en ap-northeast-1c

Los resultados son perfectos en las 2 zonas e incluso mejoran los del predictor. En la siguiente figura se muestra el coste medio obtenido en los casos de éxito en la simulación y se compara con el presupuesto fijado para la ejecución de la tarea.

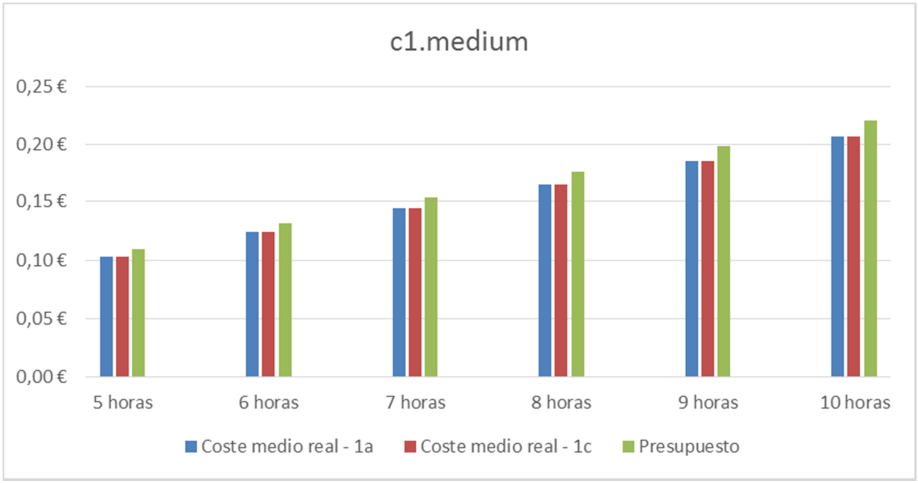


Figura 24: Coste medio real respecto al presupuesto.

En esta primera prueba tomamos una instancia cuyo precio apenas varía. La puja fijada es sólo ligeramente superior al precio que suele tener la instancia pero suficiente como para mantenerse por encima del precio en todo momento. En la tabla 9 se compara el coste medio real de los casos de éxito de cada zona con el coste que habría tenido su ejecución bajo demanda.

	5 horas	6 horas	7 horas	8 horas	9 horas	10 horas
Coste medio real - 1a	0,1032 €	0,1239 €	0,1445 €	0,1652 €	0,1858 €	0,2065 €
Coste medio real - 1c	0,1031 €	0,1238 €	0,1444 €	0,1651 €	0,1857 €	0,2064 €
Coste bajo demanda	0,5500 €	0,6600 €	0,7700 €	0,8800 €	0,9900 €	1,1000 €

Tabla 9: Comparación de coste medio real entre instancias puntuales e instancias bajo demanda**.**

Como puede apreciarse, el precio es casi idéntico en los dos casos, con diferencias por debajo del céntimo.

Pruebas con C4 2xlarge

En la figura 25 se muestra el comportamiento en la variación de precios para la instancia c4.2xlarge en la región de Asia Pacífico (Tokio). En este caso, hay elevados picos en los precios de ambas regiones.

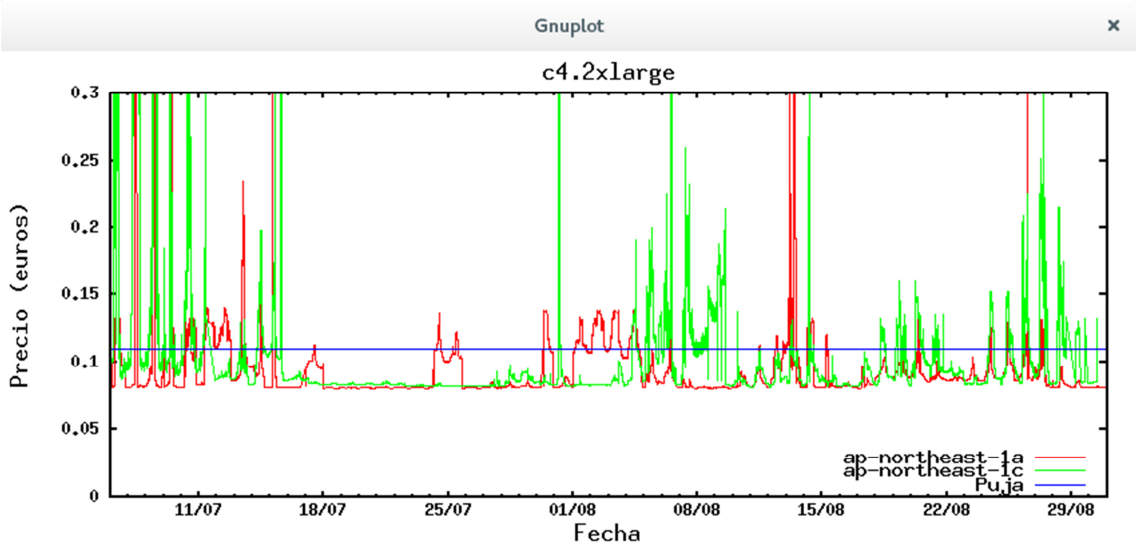


Figura 25: Histórico de precios de c4.2xlarge durante las fechas seleccionadas para el predictor.

Precio instancia bajo demanda: 0.441€/h
Puja estimada: 0.11€

COSTE TOTAL ESTIMADO	5 horas	6 horas	7 horas	8 horas	9 horas	10 horas
Coste instancia puntual	0,55€	0,66€	0,77€	0,88€	0,99€	1,1€
Coste instancia bajo demanda	2,205€	2,646€	3,087€	3,528€	3,969€	4,41€

Tabla 10: Comparación de presupuesto entre instancias puntuales e instancias bajo demanda.

Resultados del predictor

En la figura 26 se muestran los resultados del predictor para la zona *ap-northeast-1a* en tareas desde 5 a 10 horas con ambos *deadlines*.

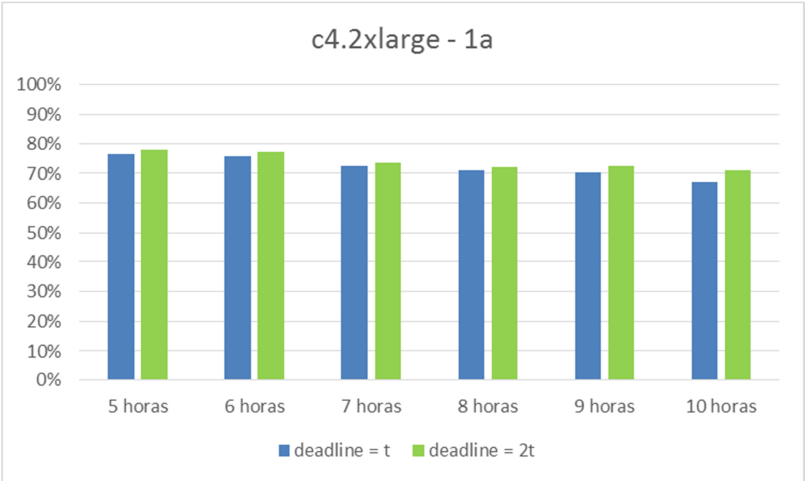


Figura 26: Resultados de la predicción en ap-northeast-1a con *deadline* = t y *deadline* = 2t

En la figura 27 se muestran los resultados del predictor para la zona *ap-northeast-1c* en tareas desde 5 a 10 horas con ambos *deadlines*.

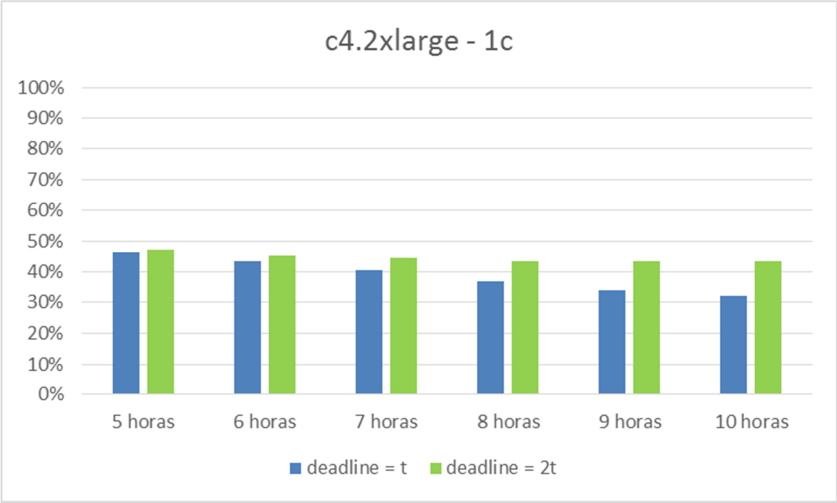


Figura 27: Resultados de la predicción en ap-northeast-1c con *deadline* = t y *deadline* = 2t

En este caso se está tratando con una máquina cuyos precios varían en mayor proporción que en la anterior. Los resultados son claramente mejores para *ap-northeast-1a*. En ambos casos, vemos como el aumento del *deadline* supone una mejora aunque apenas afecte en la puntuación final.

Resultados del simulador

En la figura 28 se muestran los porcentajes de éxito extraídos de la simulación para todas las duraciones de las tareas con *deadline* = t y *deadline* = 2t en ap-northeast-1a.

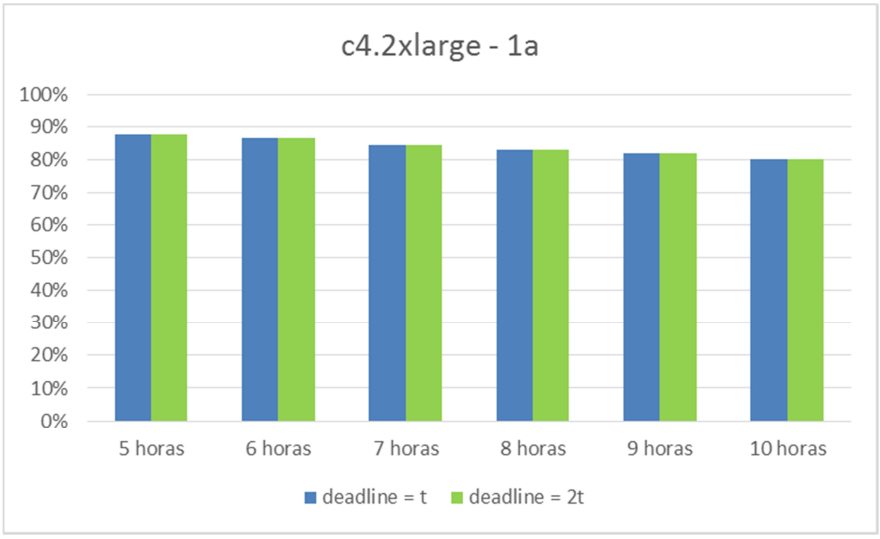


Figura 28: Resultados del simulador en ap-northeast-1a con *deadline* = t y *deadline* = 2t

En la figura 29 se muestran los porcentajes de éxito extraídos de la simulación para todas las duraciones de las tareas con *deadline* = t y *deadline* = 2t en ap-northeast-1c.

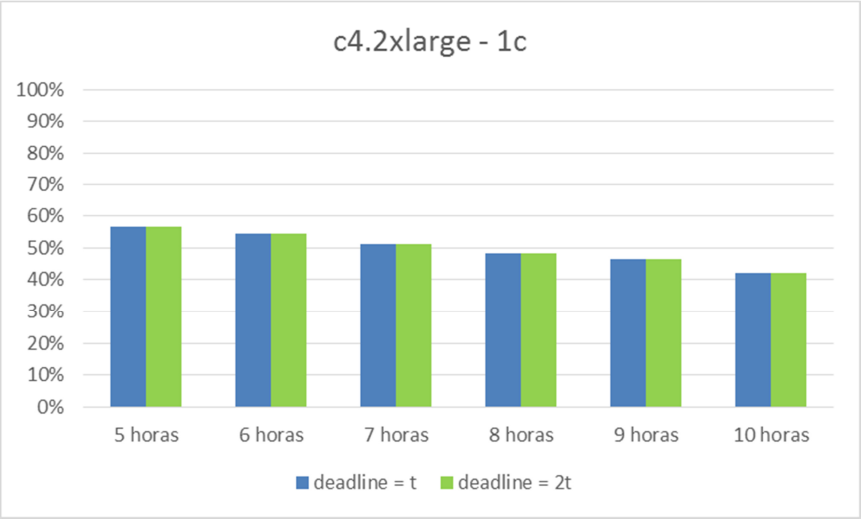


Figura 29: Resultados del simulador en ap-northeast-1c con *deadline* = t y *deadline* = 2t

Los resultados concuerdan con los del predictor. La zona ap-northeast-1a ha obtenido mejores resultados en la simulación. En ambos casos los resultados obtenidos se sitúan por encima de las predicciones. En las dos zonas se observa un aumento del 10% aproximadamente para todos los casos con respecto a la predicción inicial.

Otro hecho importante fácilmente observable es que el *deadline* no está influyendo de ningún modo en el resultado. Cuando se aumenta el *deadline* se da más margen a la tarea para comenzar. Si se observa la figura 25, se puede apreciar que los precios mantienen una estabilidad, la cual se ve alterada en ciertos momentos por subidas bastante destacadas. Por lo tanto, se puede deducir que el precio de puja que se fijó para la evaluación de esta instancia es suficiente para cubrir todos los precios en momentos de estabilidad e insuficiente ante las subidas repentinas. En definitiva, el *deadline* no influye ya que estos aumentos son los únicos momentos temporales en esta evaluación donde perdemos la instancia y no pueden salvarse con un aumento de *deadline* ya que la tarea comienza siempre de forma inmediata cuando se realiza la puja.

A continuación, en la figura 30 se muestran los costes medios en los casos de éxito. Se incluye el presupuesto para cada caso.

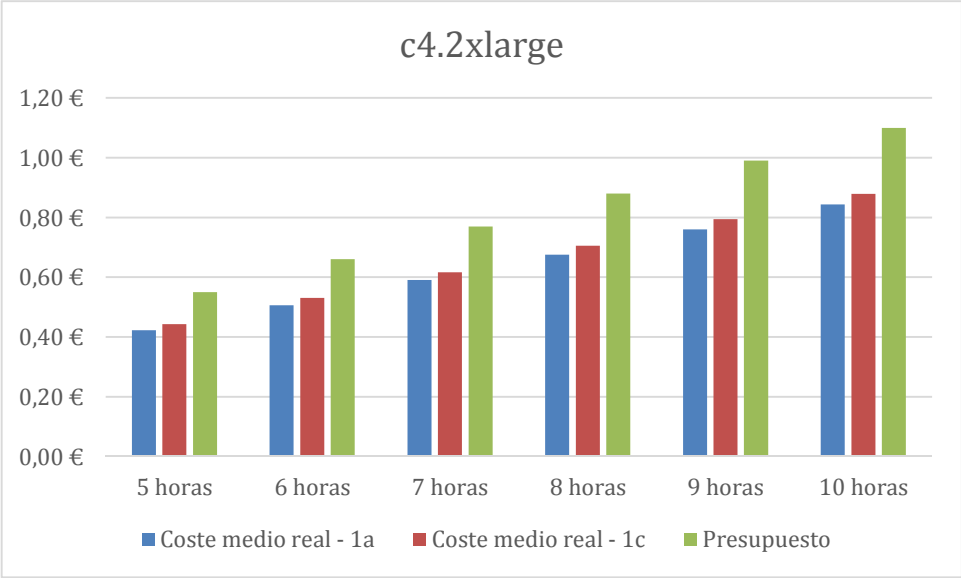


Figura 30: Coste medio real por zona para c4.2xlarge

En la tabla 11 se comparan los costes medios reales del uso de las instancias puntuales con el coste que habría tenido usando instancias bajo demanda en los casos de éxito.

	5 horas	6 horas	7 horas	8 horas	9 horas	10 horas
Coste medio real - 1a	0,42 €	0,51 €	0,59 €	0,68 €	0,76 €	0,84 €
Coste medio real - 1c	0,44 €	0,53 €	0,62 €	0,70 €	0,79 €	0,88 €
Coste bajo demanda	2,21 €	2,65 €	3,09 €	3,53 €	3,97 €	4,41 €

Tabla 11: Comparación de coste medio real entre instancias puntuales e instancias bajo demanda.

Como contrapartida a estos resultados mostramos el coste de las tareas fracasadas para cada tipo de tarea. La gráfica contempla los fracasos con coste cero. Pueden verse en la siguiente figura.

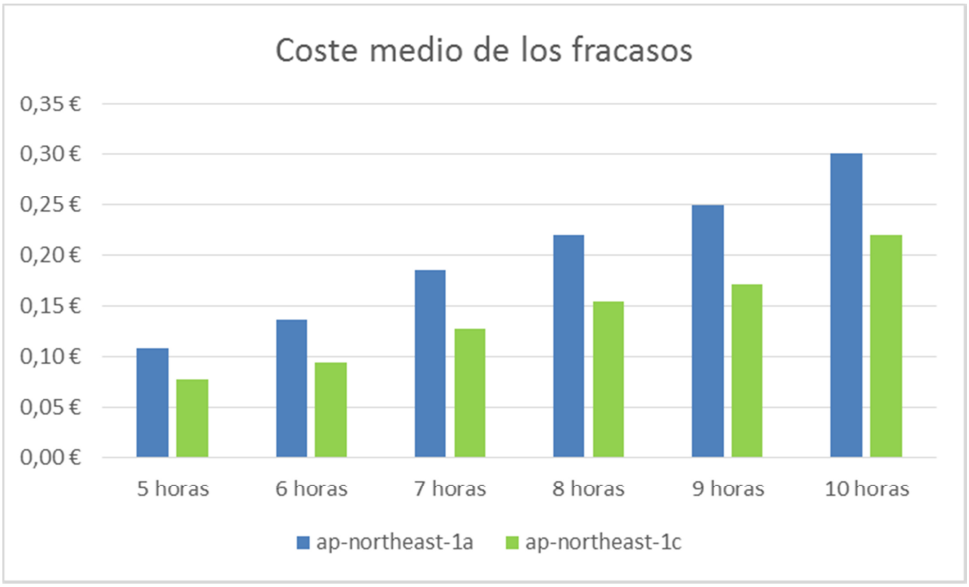


Figura 31: Coste medio de los fracasos para c4.2xlarge

Los costes por pérdidas no llegan en ningún caso a suponer el 50% del coste medio real de la tarea en el caso de ap-northeast-1a (zona con mayores costes por pérdida). Además, en *ap-northeast-1a*, el coste de los fracasos es mayor al haber comenzado la ejecución de un mayor número de tareas mientras que en *ap-northeast-1c* muchas de ellas no llegaron a completar una hora y por tanto no tuvieron coste.

Pruebas con M4 4xlarge

En este tercer tipo de instancia que se incluye en la evaluación se va a mostrar otro tipo de casuísticas diferentes que pueden surgir con el uso de las herramientas implementadas y en el comportamiento de los precios a lo largo del tiempo.

En la figura 32 se muestra el comportamiento en la variación de precios para la instancia m4.4xlarge en la región de Asia Pacífico (Tokio). En este caso, se observa una variación de comportamiento que dura varios días al inicio de la gráfica para la zona ap-northeast-1c, después que hay cierta estabilidad salvo en ciertos momentos que se dan varios picos de precios altos. La otra zona responde a un comportamiento similar, varios picos altos aunque ninguno tan exagerado como el primero nombrado.

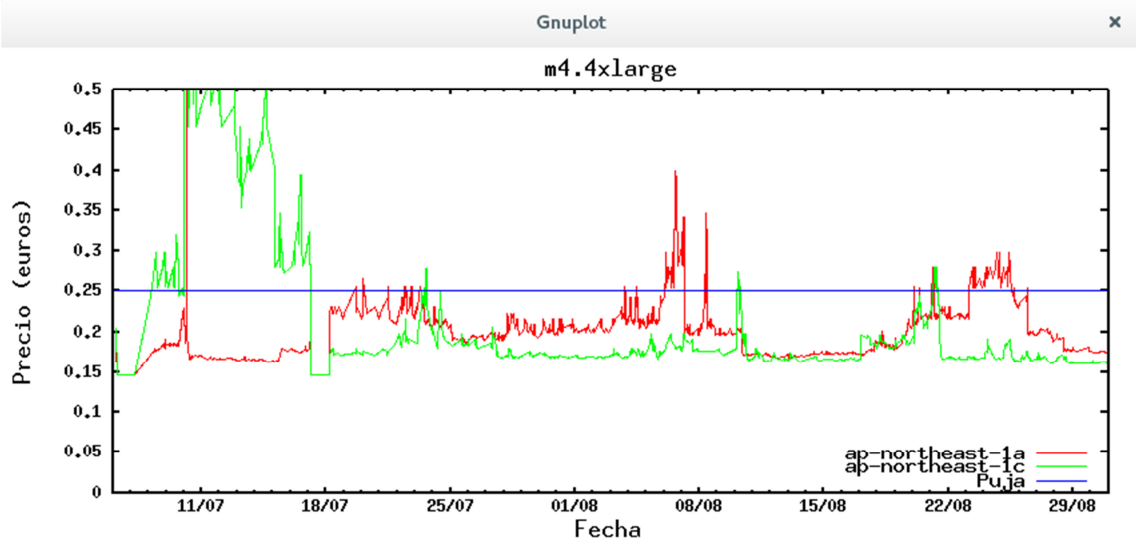


Figura 32: Histórico de precios de m4.4xlarge durante las fechas seleccionadas para el predictor.

Precio instancia bajo demanda: 0.915€/h
Puja estimada: 0.25€

COSTE TOTAL ESTIMADO	5 horas	6 horas	7 horas	8 horas	9 horas	10 horas
Coste instancia puntual	1,25€	1,5€	1,75€	2€	2,25€	2,5€
Coste instancia bajo demanda	4,575€	5,49€	6,405€	7,32€	8,235€	9,15€

Tabla 12: Comparación de coste entre instancias puntuales e instancias bajo demanda.

Resultados del predictor

En la figura 33 se encuentran los resultados del predictor para las 2 zonas posibles con los 2 *deadlines* para una tarea de 5 horas.

Para mostrar los casos concretos que se quieren tratar con esta instancia no es necesario conocer las predicciones para todas las duraciones de las tareas.

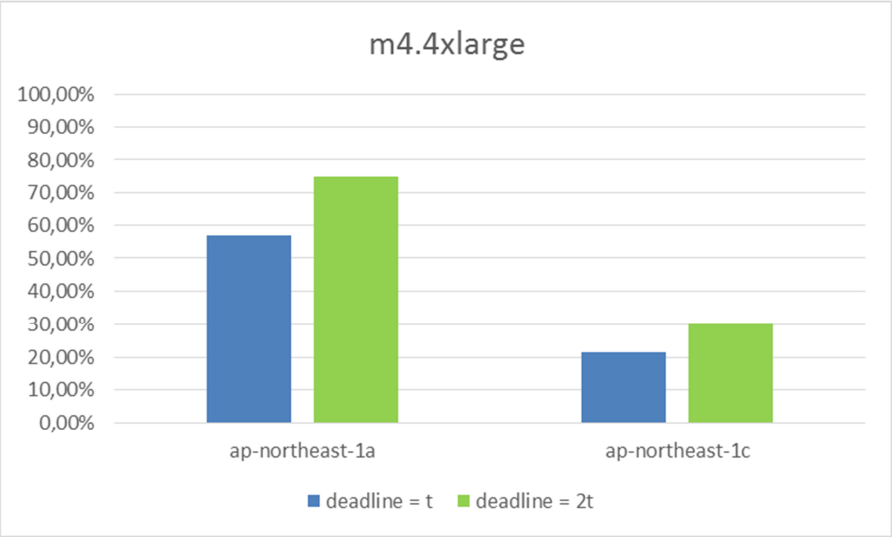


Figura 33: Resultados de la predicción en ambas zonas para t = 5 con *deadline* = t y *deadline* = 2t

Se está tratando con una máquina cuyos precios varían en mayor proporción que en las dos anteriores. Los resultados cuando el *deadline* es igual al tiempo de la tarea no son realmente buenos, en este caso es interesante valorar cómo afecta el *deadline* en la predicción en vez de centrarnos en comparar duración de tareas. Al modificar el *deadline* obtenemos un incremento cercano al 20%, situando la puntuación para ap-northeast-1a en un 75%.

La explicación ante este incremento reside en el momento inicial de la ejecución de la tarea. Como se explicó en el apartado del simulador, una vez se inicia la tarea sólo puede ser éxito o fracaso al no plantearse una nueva ejecución. Por lo tanto, cuando se fija un *deadline* igual al tiempo de la tarea, si la tarea no comienza en un primer momento ya no podrá realizarse dentro del tiempo establecido y por tanto se considera fallida. Con el incremento de *deadline* damos margen a que encuentre un momento donde el precio de puja supere el de la instancia puntual y pueda comenzar su ejecución.

Resultados del simulador

En la figura 34 se muestran los porcentajes de éxito extraídos de la simulación para todas las duraciones de las tareas con *deadline* = t y *deadline* = 2t en ap-northeast-1a.

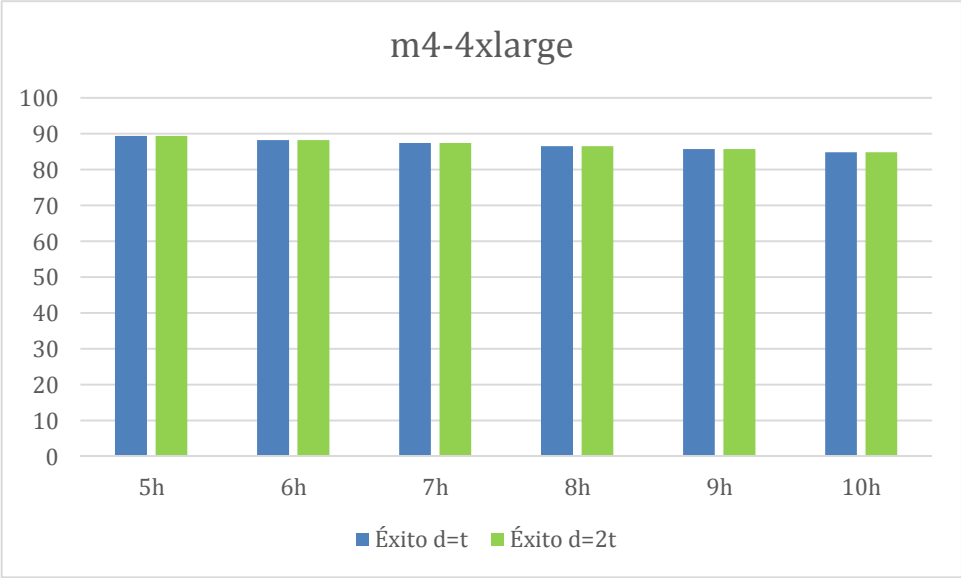


Figura 34: Resultados del simulador en ap-northeast-1a con *deadline* = 2t

En la figura 35 se muestran los resultados de la simulación para todas las duraciones de las tareas con *deadline* = t y *deadline* = 2t en ap-northeast-1c.

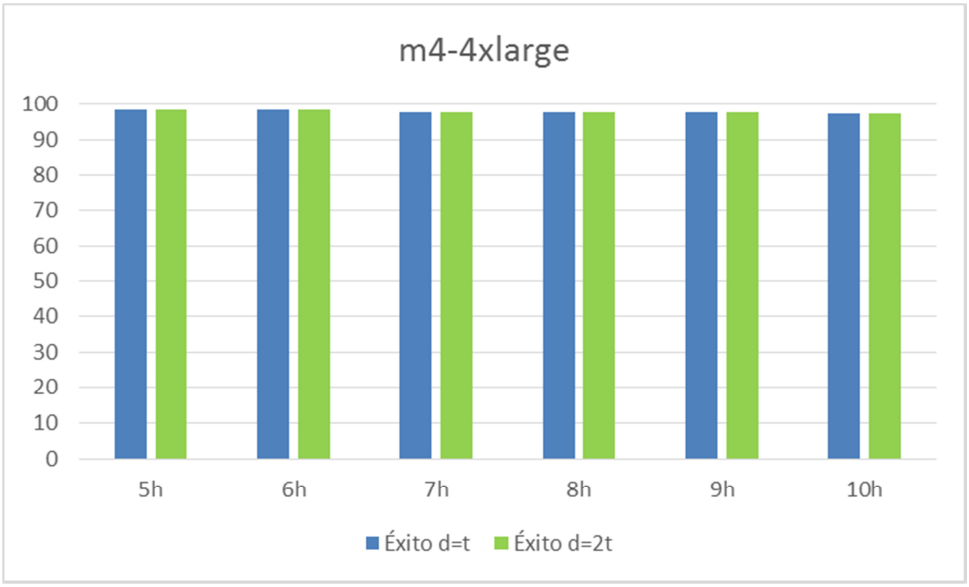


Figura 35: Resultados del simulador en ap-northeast-1c con *deadline* = t y *deadline* = 2t

Los resultados en este caso sorprenden notablemente al estar muy por encima con respecto a la predicción inicial. La diferencia podría deberse a un cambio en el comportamiento de la variación de los precios que comenzaría en torno a la última semana de la predicción. Para ello, analizamos los resultados de cada semana de forma individual.

En la siguiente figura se muestra la variación que ha sufrido la puntuación de la predicción a lo largo de las 8 semanas que analiza.

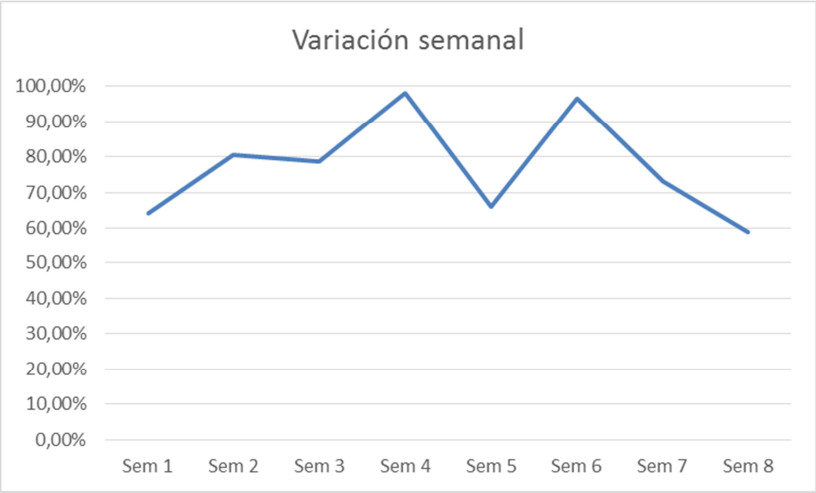


Figura 36: Variación semanal de las puntuaciones del predictor para m4.4xlarge en ap-northeast-1a

Si observamos el comportamiento a lo largo de las semanas, vemos un patrón a lo largo de ellas. Si la gráfica siguiese en las siguientes semanas con un comportamiento similar, se esperaría que justo a continuación comenzase a aumentar, lo cual se traduce en una bajada de precio (precios menores implican mayores resultados del predictor). Por lo tanto es importante destacar que no sólo hay que tener en cuenta la puntuación final sino también la variación de los precios de la instancia a lo largo del tiempo.

Se puede establecer una correspondencia de la variación semana entre los resultados de las semanas 2, 4 y 6 y la figura inicial de variación de precios. En dicha figura se observan 3 aumentos de precio en esas mismas fechas.

Como prueba definitiva de este cambio de comportamiento, se muestra en la figura 37 la variación de los precios en las fechas usadas para la simulación, donde se aprecia una bajada sustancial del precio de la instancia pudiendo mantener el valor de la puja por encima en la mayoría de instantes.

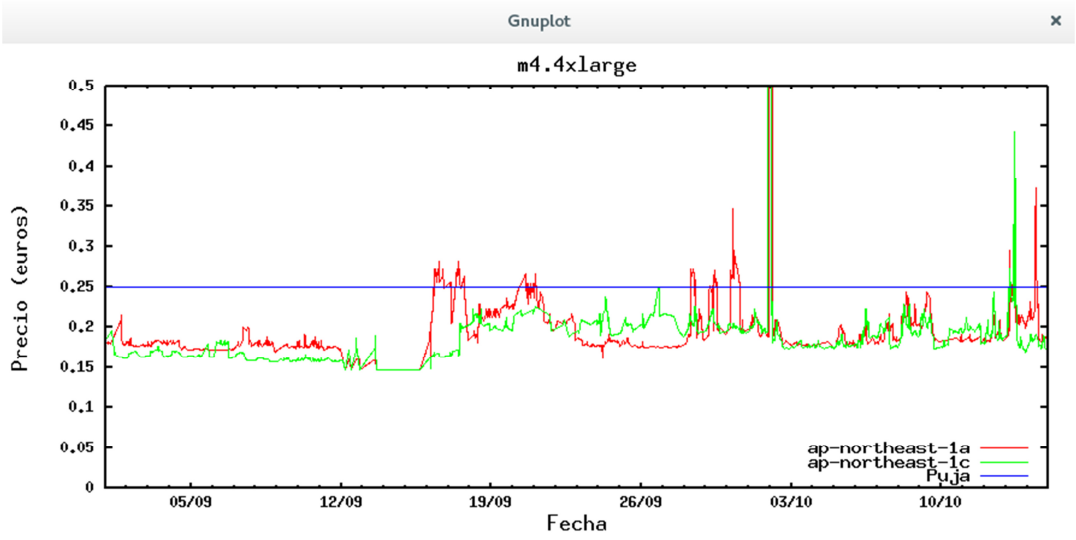


Figura 37: Variación del precio de los datos usados en la simulación para m4.4xlarge

Esta bajada de precio en la instancia implica que se puede ajustar todavía más el precio de la puja para aumentar el ahorro con respecto a las instancias bajo demanda. En la figura 38 se toma de punto de partida una tarea de 5 horas y se muestra cómo varía su porcentaje de éxitos conforme se disminuye el precio de puja.

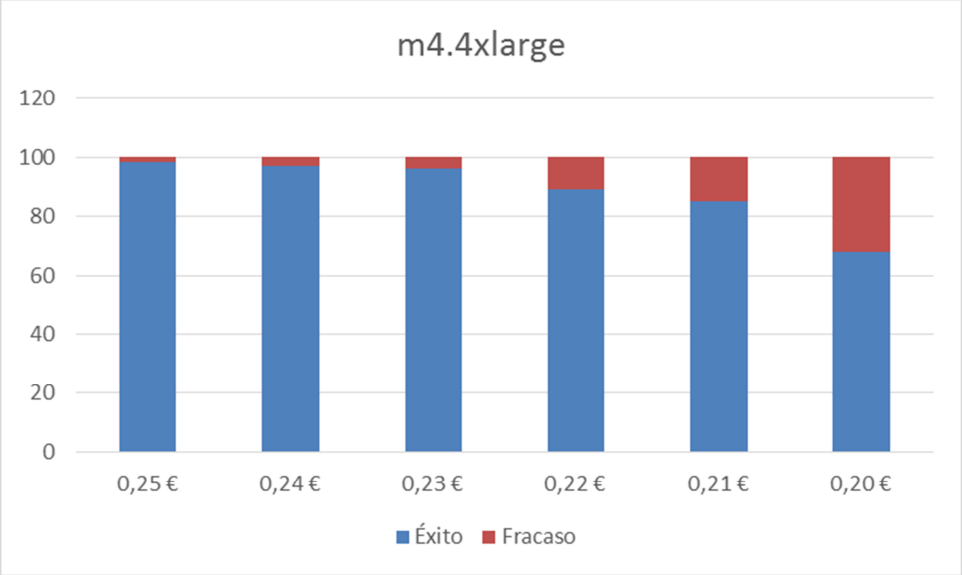


Figura 38: Variación de porcentajes de éxito/fracaso en tareas de 5h con la disminución del precio de puja en ap-northeast-1c.

Como puede verse en la figura 38, la bajada de precios esperada en las siguientes semanas a las usadas en la predicción nos permitiría ajustar los precios para obtener un rendimiento mayor del presupuesto fijado. Podría considerarse admisible una bajada hasta los 0,22€ al mantener un 90% de casos de éxito.

A continuación, en la figura 39 se muestran los costes medios en los casos de éxito. Se incluye el presupuesto para cada caso.

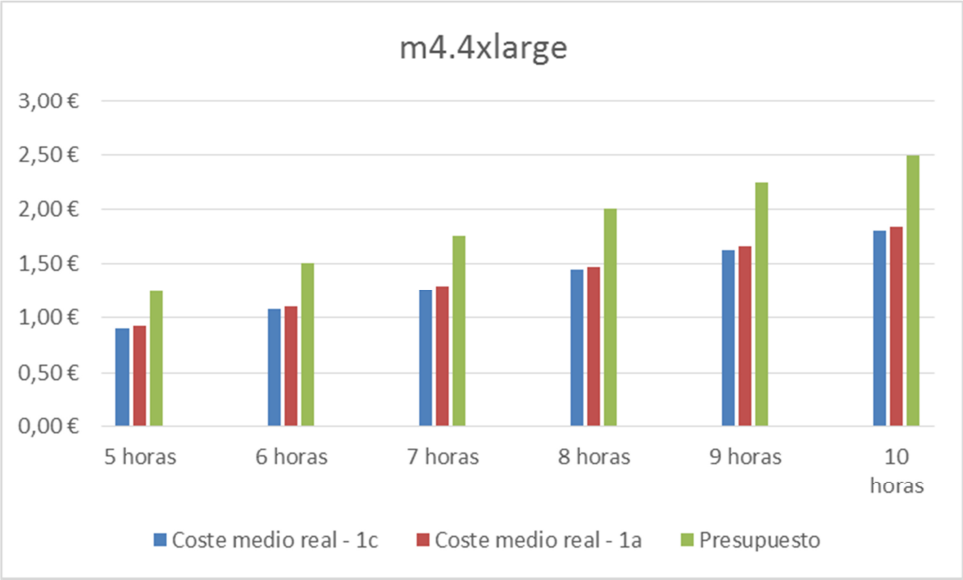


Figura 39: Coste medio real por zona para m4.4xlarge

En la tabla 13 se comparan los costes medios reales del uso de las instancias puntuales con el coste que habría tenido usando instancias bajo demanda en los casos de éxito.

	5 horas	6 horas	7 horas	8 horas	9 horas	10 horas
Coste medio real - 1c	0,90 €	1,08 €	1,26 €	1,44 €	1,62 €	1,80 €
Coste medio real - 1a	0,92 €	1,10 €	1,29 €	1,47 €	1,66 €	1,84 €
Coste bajo demanda	4,58 €	5,49 €	6,41 €	7,32 €	8,24 €	9,15 €

Tabla 13: Comparación de coste medio real entre instancias puntuales e instancias bajo demanda

Como contrapartida a estos resultados mostramos el coste de las tareas fracasadas para cada tipo de tarea. La gráfica contempla los fracasos con coste cero. Pueden verse en la siguiente figura.

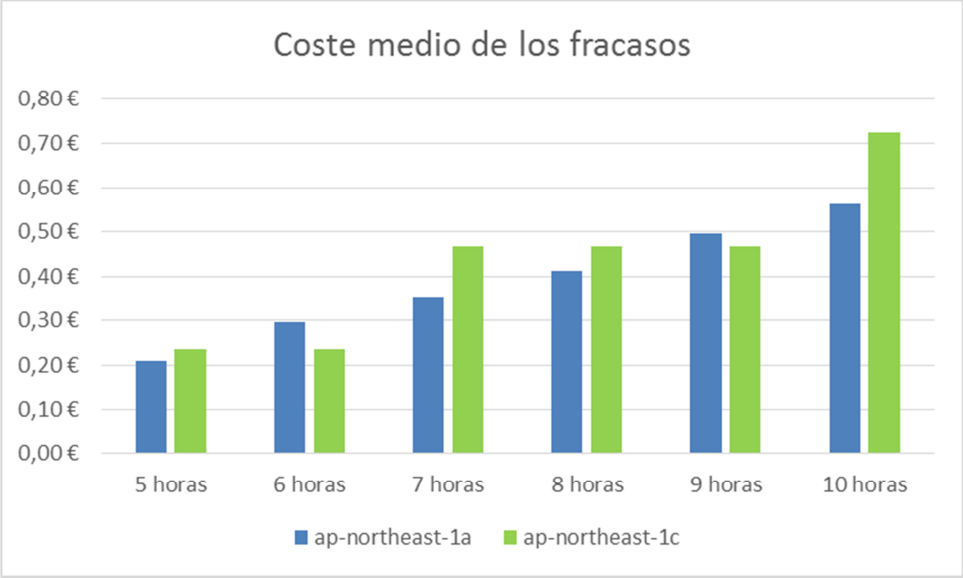


Figura 40: Coste medio de los fracasos para m4.4xlarge

Se observa que en ningún caso llegan a suponer un 50% del coste medio por tarea usando instancias puntuales. Si comparamos estas cifras con las instancias bajo demanda el porcentaje disminuye hasta un 10% del coste de la instancia bajo demanda. Esta pequeña cantidad en coste que supondría un fracaso junto con el porcentaje de éxito obtenido en las pruebas hacen de esta instancia una buena opción para la ejecución de tareas de las duraciones propuestas.

5.2. Evaluación de la solución

Durante la evaluación se han visto diferentes tipos de instancias cuyos precios tenían comportamientos dispares. Tras el uso del predictor, el simulador y evaluar las casuísticas particulares que han ido surgiendo, se establecen las siguientes conclusiones para cada uno de los tipos de instancias analizadas.

Existen instancias con precios casi constantes (como c1.medium) donde el precio, por normal general, no experimenta cambios bruscos. Son instancias que nos permiten ajustar al máximo el precio sin que ello conlleve un riesgo elevado. En este grupo de máquinas se encuentran las que tienen los precios más bajos. Las instancias con precios casi constantes.

A continuación, se comenta brevemente los resultados para cada tipo de instancia:

- c1.medium: Caso más básico, instancia con precio apenas variable. Los resultados de la simulación resultaron algo previsibles y se ajustaron bien a los del predictor.
- c4.2xlarge: Instancia cuyos precios varían. A pesar de la variación, el precio solía mantenerse dentro de un rango buena parte del tiempo. Destacan en este caso la cantidad de picos que pueden darse. Los resultados de la simulación corresponden con los de la predicción.
- m4.4xlarge: Instancia cuyos precios de forma similar al caso de la instancia c4.2xlarge. Este caso particular experimentó un cambio de comportamiento en los precios justo en las fechas correspondientes al final de la predicción y al comienzo de la simulación. No hubo relación en los resultados. Se averiguó que es necesario estudiar la tendencia de los precios e introducirlo en la valoración para aportar un mayor valor a la puntuación final.

Consideramos que la solución cumple con su cometido siempre y cuando no se experimente una variación en la tendencia general de los precios.

6. Conclusiones y trabajo futuro

En este apartado se detallan las conclusiones sobre el proyecto, las posibles vías de trabajo futuro que podrían tomarse a raíz del estudio realizado y la valoración personal del mismo.

6.1. Conclusiones

Tras la evaluación realizada en el punto anterior, consideramos que el objetivo del proyecto está cumplido. Se ha conseguido desarrollar un sistema capaz de evaluar la viabilidad de ejecutar aplicaciones científicas utilizando las instancias puntuales de Amazon, se ha mostrado que esta alternativa puede suponer un elevado ahorro de coste y se ha evaluado la utilidad de las herramientas desarrolladas. En la evaluación se han presentado tres casos diferentes que nos han permitido establecer diferentes conclusiones.

En el primer experimento no hubo lugar a dudas, las instancias puntuales con precios bajos son las que más estables se mantienen, esto las hace perfectas para garantizar la máxima disponibilidad en casi todos los casos.

En el segundo experimento los resultados fueron muy positivos. A pesar de ser una instancia variable los resultados entre la predicción y el simulador se correspondían entre sí y se logró un buen porcentaje de éxitos.

En el tercer experimento se observó un comportamiento anormal. Tras analizarlo se pudo concluir que era necesario establecer otro criterio en la valoración (comportamiento del precio a lo largo del tiempo) de la viabilidad de las tareas. Si para instancias de este tipo se tiene en cuenta este nuevo criterio, se obtendrán resultados similares a los del segundo experimento.

6.2. Trabajo futuro

Tras la finalización del proyecto considero que hay múltiples vías por las que continuar el trabajo así como mejoras sobre el mismo.

Consideración de la variabilidad del precio en el cálculo de la viabilidad: La herramienta de predicción debería considerar la variabilidad de los datos correspondientes a diferentes semanas con el objetivo de mostrar resultados más precisos en relación a los determinados por el simulador.

Evaluación de diferentes vectores de peso: Se podrían evaluar diferentes vectores de peso con el objetivo de analizar si es posible obtener una mayor correlación entre los

resultados del predictor y los del simulador. Del mismo modo se podría variar el número de semanas utilizado para calcular la viabilidad de ejecutar una tarea.

Mecanismos de puja dinámicos y autónomos: Hacer que la puja varíe en función del estado de ejecución de la tarea. Cuestiones que se plantean (por ej.):

- ¿Queremos aumentar la puja si nos acercamos al *deadline* y todavía no empezamos la tarea?
- ¿Compensa aumentar la puja en las horas finales de la ejecución de una tarea para asegurar una mayor disponibilidad de los recursos?
- La tarea ha fallado pero sólo me han cobrado 1 hora. ¿Podría ejecutarse todavía la tarea con el presupuesto restante en el *deadline* establecido? De ser así podría modificarse la puja en base a la nueva puntuación obtenida.

Mejoras en disponibilidad mediante migración de tareas: El comportamiento de los precios en una zona o en un determinado tipo de máquina no nos asegura la finalización de las tareas. ¿Podrían entrar en juego varias zonas y migrar dichas tareas de una zona a otra en determinados momentos? En este apartado entraría también todo el tema del *checkpointing*, guardar el estado de ejecución de una tarea y restaurarla en otra instancia.

Definición de mecanismos de aprendizaje en base al resultado de una puja: El sistema podría almacenar si la puja realizada ha tenido éxito o ha fracasado y utilizar esa información para decidir el precio de pujas posteriores.

Relanzamiento automático de tareas: El sistema podría considerar el relanzamiento de las tareas que fracasan con el objetivo de intentar finalizar la misma antes del *deadline* fijado. Para ello, se podría usar la misma estrategia restando del presupuesto la cantidad gastada por la tarea hasta ese momento.

6.3. Valoración personal

Personalmente considero que se trataba de un proyecto un tanto diferente y que no habría tenido oportunidad de hacer en otras circunstancias. Lo que más destacaría del proyecto es que no se trataba de una mera implementación en sí, sino que iba más enfocado al estudio y planteamiento de una temática desconocida para mí.

Me pareció interesante el tener que plantear todo desde el paso inicial de documentación hasta la definición final de la solución. Considero que el partir del trabajo y estudio previo realizado por otras personas me ha dado una visión del proceso que antes no tenía.

El uso de herramientas con las que no había trabajado hasta el momento (AWK, Gnuplot) siempre es positivo, me facilitaron ciertas tareas como el procesado de ficheros y la generación de gráficas cuando se manejan grandes volúmenes de datos (miles).

A pesar de que considero que me defiendo bien con el inglés la lectura de los estudios sobre el tema no resultó sencilla, por la densidad del contenido en sí y la terminología en algunos de ellos que además hacían un uso bastante extenso de las matemáticas.

ANEXO I: Implementación

En este anexo se detalla cómo se llevó a cabo la implementación de cada uno de los apartados que componen el sistema.

En la figura I.1 se muestra el esquema general de todos los sistemas implementados. En el esquema pueden distinguirse tres componentes claramente diferenciados. Se pasan a explicar brevemente estos componentes.

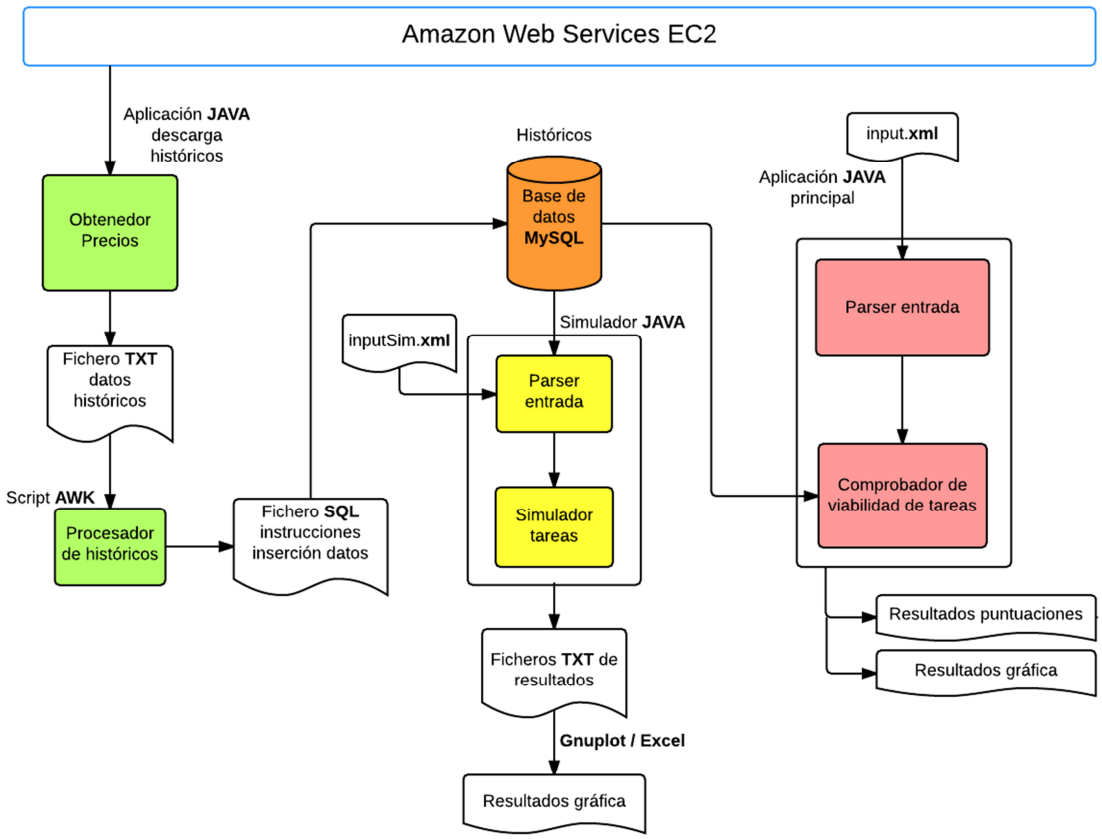


Figura I.1: Esquema general del sistema

Aplicación principal: Aplicación JAVA que se encarga de valorar la viabilidad de las tareas y de mostrar al usuario estas valoraciones. Recibe como entrada un fichero XML con la especificación de la tarea y muestra el resultado a través de la salida estándar de texto así como en una gráfica.

Aplicación de descarga de históricos: Aplicación JAVA que solicita los históricos de precios a los servicios web de Amazon y los traslada a un fichero de texto.

Simulador: Aplicación JAVA utilizada para simular tareas de diverso tipo. Recibe los datos de la tarea a través de un fichero XML y traslada los resultados a un fichero de texto.

Todos estos componentes interactúan en torno a una **base de datos** MySQL.

I.1. Creación de una base de datos actualizada y autónoma.

En este apartado se detalla la implementación del componente que gestiona la base de datos que contiene los históricos de precios de las instancias puntuales de Amazon. Esta base de datos se actualiza de forma automática con los nuevos datos que se van generando en cada momento y es utilizada por varios de los componentes del sistema como son el simulador y la aplicación principal de valoración de viabilidad de tareas.

En la figura I.2 se muestra el esquema de la base de datos autónoma y su interacción con otros componentes:

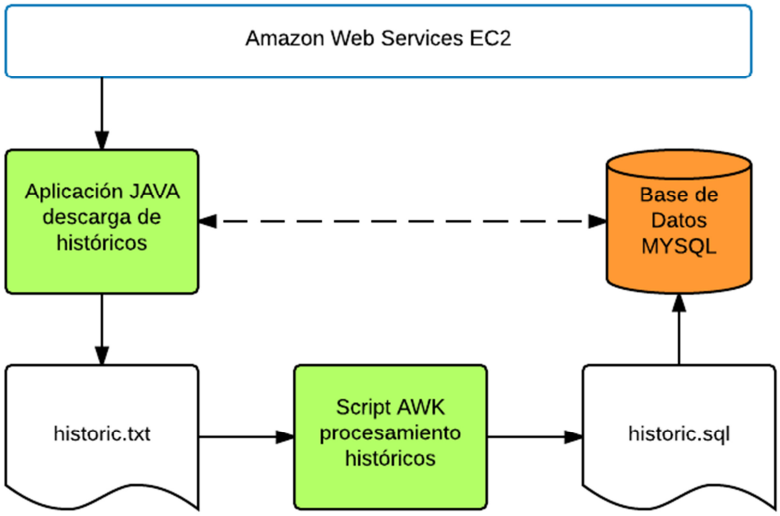


Figura I.2: Esquema de interacción de la base de datos.

I.1.1. Motivos del uso de la base de datos

La aplicación principal (predicción de viabilidad) necesita datos de históricos que la propia Amazon nos proporciona. Dicha información puede obtenerse a través de sus servicios web. La descarga de la información a través de estos servicios puede ser algo ineficiente por cuestiones de coste temporal.

Otro punto a tener en cuenta es que los históricos tienen información sobre los últimos dos meses como máximo y es posible que en el futuro nos interese utilizar más datos.

De los dos puntos anteriores deducimos que lo mejor en este caso es la creación de una base de datos que de persistencia a esta información garantizándonos un acceso

mucho más rápido además de poder almacenar históricos por más de dos meses. Para su implementación se usó MySQL 5.6.

I.1.2. Estructura de la base de datos

La base de datos cuenta con tablas para almacenar todos los precios de todas las máquinas de cada zona. Es importante tener en cuenta que Amazon facilita unos históricos que contienen las variaciones de los precios y no el precio que tenían en cada momento. De este modo la información requerida es mucho menor. Los nombres de dichas tablas siguen la siguiente estructura:

tipo_tamaño_region

Con esta nomenclatura de tablas es fácil identificar las características de la instancia sobre la que guardan información así como su localización geográfica. Podría almacenarse todo en una misma tabla ya que el contenido en todas ellas es el mismo pero de este modo es necesario recorrer un número de datos mucho menor resultando realmente sencilla su implementación.

Algunos ejemplos de nombres:

- C1_MEDIUM_US
- C1_MEDIUM_UE
- M1_XLARGE_SA
- R3_8XLARGE_AP

Hay que tener en cuenta que en cada una de las regiones existen varias zonas. Por ejemplo en los centros de datos de Estados Unidos podemos encontrar las zonas “us-west-2a” y “us-west-2b” entre otras, la información de ambas zonas se concentrará en una única tabla para cada tipo de máquina.
La información que se guarda en las tablas es la siguiente:

- *Zone* : zona específica en la que se encuentra la instancia.
- *Description*: describe el sistema que utiliza la instancia. Existen varios tipos, también disponibles como *Virtual Private Cloud* (VPC, [W1]):

.1 Linux/UNIX
.2 SUSELinux
.3 Windows

- *Price*: precio de la instancia. Hay que tener en cuenta que el precio es válido hasta que otro registro de la misma instancia, zona y descripción aparezca en la base de datos. Esto es así ya que los registros de Amazon no almacenan el valor para cada hora/minuto/segundo sino que contiene las variaciones del precio (resultando mucho más eficiente en coste y almacenamiento.
- *Time*: fecha del registro.

- *WeekDay*: día de la semana correspondiente a la fecha del campo “Time”.
- *TimeZone*: zona horaria, puede ser CET o CEST (en el caso que nos ocupa) ya que Amazon detecta desde donde se está efectuando la petición y da las fechas transformadas a nuestra zona horaria. De otro modo habría que realizar la conversión para ver qué horas de otros continentes corresponden con las horas que nos interesan.

A continuación, la tabla I.1 muestra un código SQL de ejemplo para la creación de una de las tablas que almacenan los históricos de las instancias:

```
CREATE TABLE IF NOT EXISTS c1_medium_AP
(
        id              BIGINT NOT NULL AUTO_INCREMENT,
        Zone            VARCHAR(20) NOT NULL,
        Description     VARCHAR(40) NOT NULL,
        Price           FLOAT NOT NULL,
        Time            TIMESTAMP NOT NULL,
        WeekDay         VARCHAR(5) NOT NULL,
        TimeZone        VARCHAR(5) NOT NULL,
        PRIMARY KEY (id)
);
```

Tabla I.1: Ejemplo de código SQL de creación de tabla de instancia

Puede observarse que aparece un nuevo campo “id”. Este campo sirve para diferenciar cada uno de los registros. En un principio se pensó establecer como clave primaria una clave compuesta por los campos “Zone”, “Description” y “Time” ya que la información sobre el tipo de instancia ya está patente en el propio nombre de la tabla y por tanto dicha clave nos daría como resultado registros únicos, no podría haber 2 precios para una misma instancia de la misma zona y tipo en la misma fecha. El problema era que este tipo de claves son realmente ineficientes y no aportaban nada realmente útil en las aplicaciones que iban a hacer uso de la base de datos, por lo tanto se descartó su uso.

A continuación, la figura I.3 muestra un ejemplo del contenido de las tablas:

id	Zone	Description	Price	Time	WeekDay	TimeZone
54865	us-west-2b	Linux/UNIX	0.2563	2014-06-30 10:00:47	Mon	CEST
54857	us-west-2a	Linux/UNIX	0.2579	2014-06-30 10:26:48	Mon	CEST
54837	us-west-2a	Linux/UNIX	0.2568	2014-06-30 10:33:10	Mon	CEST
54838	us-west-2c	Linux/UNIX	0.2573	2014-06-30 10:33:10	Mon	CEST
54823	us-west-2a	Linux/UNIX	0.2578	2014-06-30 10:37:23	Mon	CEST
54819	us-west-2c	Linux/UNIX	0.2574	2014-06-30 10:39:27	Mon	CEST
54817	us-west-2c	Linux/UNIX	0.2573	2014-06-30 10:41:30	Mon	CEST
54805	us-west-2a	Linux/UNIX	0.2569	2014-06-30 10:49:48	Mon	CEST

Figura I.3: Contenido de las tablas de instancias

I.2. Actualizador

El “actualizador” no es un componente del sistema en sí sino un conjunto de ellos que funcionan de manera coordinada. En la figura I.4 se presentan todas las partes que integran este actualizador.

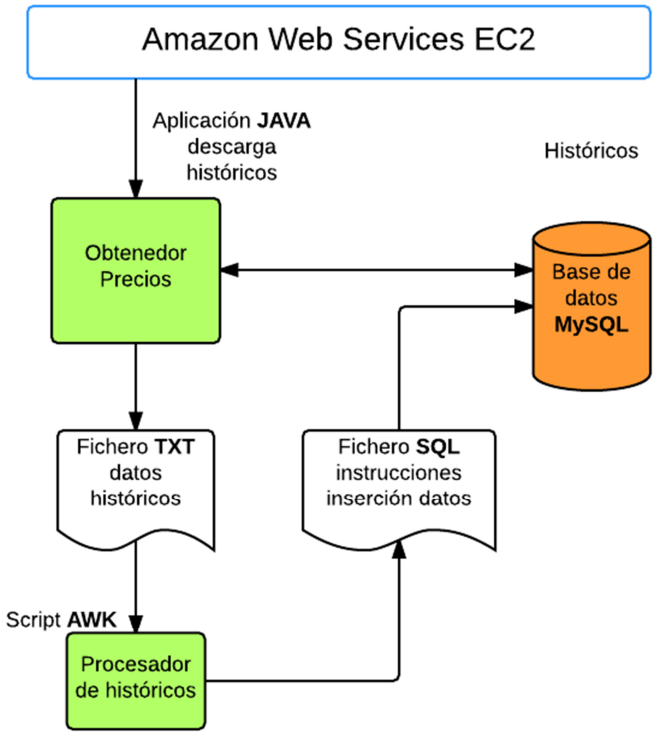


Figura I.4: Componentes que integran el actualizador

En los siguientes apartados se explican de forma individual así como su integración conjunta.

I.2.1. Aplicación JAVA para la obtención de los históricos de precios

La función de la aplicación es la de solicitar a los servicios de Amazon el histórico de precios de todas las zonas de las que dispone hasta la fecha de la última actualización realizada en nuestra base de datos. En este punto entra en juego una tabla extra que se decidió incorporar a la base de datos. Dicha tabla contiene las fechas de las últimas actualizaciones realizadas hasta el momento en cada zona.

La información que se guarda en la tabla es la siguiente:

- *Zone*: región de Amazon.
- *Time*: fecha de la última actualización.

A continuación, la tabla I.2 muestra el código SQL para la creación de la tabla que contiene las fechas de las últimas actualizaciones:

```
CREATE TABLE zone_updates (
        Zone            VARCHAR(20) NOT NULL,
        Time            TIMESTAMP NOT NULL,
        PRIMARY KEY (Zone)
);
```

Tabla I.2: Código SQL de la tabla de actualizaciones

En la figura I.5 puede verse el contenido completo de la tabla, en ella se pueden observar todas las zonas que se utilizan en la aplicación y en la base de datos.

Zone	Time
AP_NORTHEAST_1	2014-12-15 11:22:03
AP_SOUTHEAST_1	2014-12-15 11:22:49
AP_SOUTHEAST_2	2014-12-15 11:22:43
EU_CENTRAL_1	2014-12-15 11:08:22
EU_WEST_1	2014-12-15 10:57:21
SA_EAST_1	2014-12-15 15:49:35
US_EAST_1	2014-12-15 11:15:40
US_WEST_1	2014-12-15 11:17:21
US_WEST_2	2014-12-15 11:18:04

Figura I.5: Ejemplo de contenido de la tabla de actualizaciones

Para interactuar con los servicios de Amazon se ha optado por usar una aplicación Java y el SDK que facilitan para dicho lenguaje y que es completamente integrable en Eclipse (IDE utilizado) [W3].

A continuación se describe la ejecución de la aplicación:

- Para cada una de las zonas:

.1 Obtiene última fecha de actualización.

.2 Fija la última fecha de actualización como fecha límite hasta la que descargar el histórico.

.3 Descarga el histórico y lo añade a un fichero de texto.

.4 Actualiza la fecha de actualización hasta la más reciente obtenida.

I.2.2. Procesamiento de los históricos de precios mediante scripts en AWK

Los ficheros de texto que contienen la información de los históricos no pueden introducirse directamente en la base de datos y es necesario procesarlos. El tamaño de los ficheros es bastante grande pudiendo sobrepasar el millón de líneas (hay que tener en cuenta además que son 8 zonas). Para este fin se usó un script en AWK ya que es realmente rápido para procesar textos. El script se encarga de trocear cada registro del histórico y de generar una sentencia SQL para la base de datos especificada (sentencia

de inserción en este caso). El resultado final de la ejecución del script es un fichero SQL llamado “dataSQL.sql”.

La tabla I.3 muestra el código del script utilizado:

```
#! /bin/awk -f
# Programa awk que transforma el fichero de logs en sentencias sql
# de inserción de datos para la base de datos pasada por parámetros.
# Ejemplo de ejecución:  awk -v bd=tfg -f generator.awk log4.txt

BEGIN {
        FS = "[:|,|{|}]"
}

{
        #Asigno variables y elimino espacios con "sub"
        table=$3
        desc=$5
        sub(" ","",desc);
        price=$7
        sub(" ","",price);
        zone=$13
        sub(" ","",zone);
        weekday=substr($9,2,3);
        year=substr($11,8,5);
        month=substr($9,6,3);
        day=substr($9,10,2);
        hour=substr($9,13,2);
        minute=$10
        gsub(" ","",minute);
        second=substr($11,1,2);
        timezone=substr($11,4,4);

        #Transformar mes
        if (month == "Jan") {
                month=01;
        }
        else if (month == "Feb") {
                month=02;
        }
        else if (month == "Mar") {
                month=03;
```

```
}
else if (month == "Apr") {
        month=04;
}
else if (month == "May") {
        month=05;
}
else if (month == "Jun") {
        month=06;
}
else if (month == "Jul") {
        month=07;
}
else if (month == "Aug") {
        month=08;
}
else if (month == "Sep") {
        month=09;
}
else if (month == "Oct") {
        month=10;
}
else if (month == "Nov") {
        month=11;
}
else if (month == "Dec") {
        month=12;
}

#Generar nombre tabla
if ($13 ~ "eu"){
        table=$3"_EU"
}
else if ($13 ~ "us"){
        table=$3"_US"
}
else if ($13 ~ "ap"){
        table=$3"_AP"
}
else if ($13 ~ "sa"){
        table=$3"_SA"
```

```
        }

        #Modifico el nombre de la tabla para cambiar el "." por un "_" y quitar un
espacio
        sub(" ","",table);
        sub(/\./,"_",table);


        sub(" ","",year);
        sub(",","",year);

        sub(" ","",timezone);

        #Construyo la fecha
        date=year"-"month"-"day" "hour":"minute":"second

        print "INSERT INTO "bd"."table"
(Zone,Description,Price,Time,WeekDay,TimeZone) VALUES
(\""zone"\",\""desc"\","price",'"date"',\""weekday"\",\""timezone"\");" >
"dataSQL.sql"
}
```

Tabla I.3: Código AWK del script de procesado de históricos

I.2.3. Automatización de las actualizaciones de la base de datos

Es necesario que la base de datos esté siempre actualizada ya que es uno de los requisitos de la aplicación principal de predicción. Si no lo está su comportamiento podría no ser el esperado. Además con la automatización conseguimos que el usuario deje de preocuparse por el estado de la base de datos, siempre contará con información actualizada pudiendo también contar con los registros más antiguos en caso de querer realizar otro tipo de comprobaciones.

Llegados a este punto tenemos 3 partes claramente diferenciadas:

- Base de datos para dar persistencia a la información.
- Programa Java que descarga los históricos.
- Script AWK que procesa los históricos y genera un fichero SQL.

La base de datos se actualiza diariamente. Se optó por actualizaciones diarias ya que el tiempo que requería no era especialmente grande y ampliar el tiempo entre actualizaciones a intervalos de una semana o más produciría errores en la aplicación. Es necesario que esté actualizada hasta una semana atrás como mínimo.

Para automatizar el proceso se programó el cron para que cada día ejecutase un script que realizase los siguientes pasos en este orden:

- La aplicación de descarga de históricos.
- El script awk de procesado de históricos y creación de fichero SQL (para todos los ficheros descargados)
- Ejecución de todos los SQL generados.
- Borrado de los ficheros usados.

Configuración inicial del equipo:

- Debian 8.2.0
- Java 7
- MySQL 5.6

A continuación se presenta el script utilizado:

```
#!/bin/bash
javapath="/home/sanz/Escritorio/tfg/BD_Updater/java/"
parserpath="/home/sanz/Escritorio/tfg/BD_Updater/parser"

# Launch java program to obtain log info
Java -cp
".:/home/sanz/Escritorio/tfg/BD_Updater/java:/home/sanz/Escritorio/tfg/BD_Updater
/lib/*" PricesClient.ObtenedorPrecios localhost tfg root root

#Process Log
mv /home/sanz/AP_NORTHEAST_1.txt $parserpath/tmp/AP_NORTHEAST_1.txt
awk -v bd=tfg -f $parserpath/generator.awk $parserpath/tmp/AP_NORTHEAST_1.txt
cat $parserpath/tmp/tables.txt | sort | uniq > $parserpath/tmp/tables2.txt
awk -f $parserpath/generateTables.awk $parserpath/tmp/tables2.txt

rm $parserpath/tmp/tables.txt
rm $parserpath/tmp/tables2.txt

# Process finished: launch mysql script for tables
mysql -u root -proot tfg < $parserpath/tmp/tables.sql
mysql -u root -proot tfg < $parserpath/tmp/dataSQL.sql

#Delete SQL files
rm $parserpath/tmp/tables.sql
rm $parserpath/tmp/dataSQL.sql

#Delete zone files
rm $parserpath/tmp/AP_NORTHEAST_1.txt
```

Tabla I.4: Código del script que ejecuta el cron

En este caso el script es para una región, para llevarlo a otra sólo habría que duplicarlo.

Para automatizar el script sólo hay que incluir la siguiente entrada en el fichero */etc/crontab*:

0 4 * * * sanz /bin/bash /home/sanz/Escritorio/tfg/updateDB.sh

Así queda configurado para su ejecución a las 4 de la mañana todos los días.

NOTA: Las rutas que figuran en el script están adaptadas a la estructura de directorios y ficheros que tenía en el sistema.

I.3. Simulación de tareas

Para simular si una tarea se habría ejecutado correctamente dado un precio de puja, contamos con una herramienta desarrollada en Java que simularía el proceso de puja que se llevaría a cabo. El motivo de la creación de dicha herramienta es el de poder ampliar el espectro de pruebas que somos capaces de realizar sin las limitaciones que supondrían en tiempo y coste económico. En la figura I.6 se muestra el esquema general de este componente.

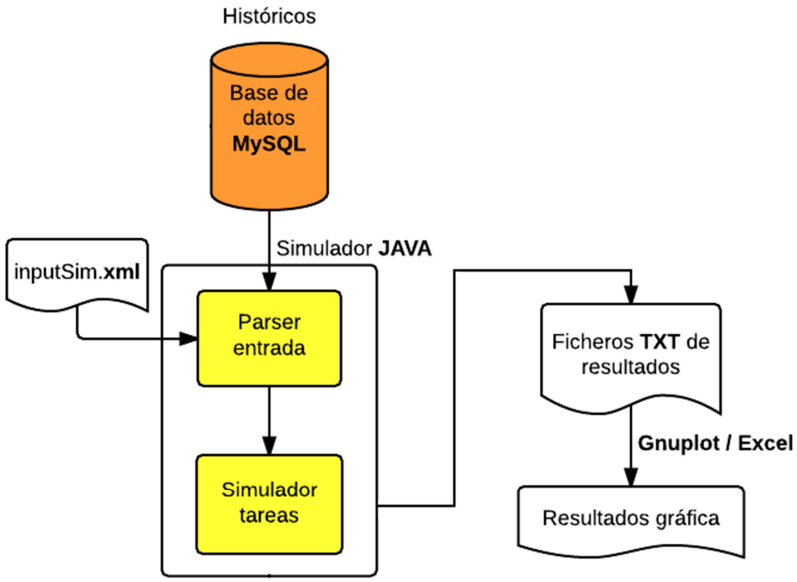


Figura I.6: Esquema general del simulador

El simulador recibe la tarea a simular desde un fichero XML. Procesa la entrada y simula la tarea. Las simulaciones las hace únicamente con datos contenidos en la base de datos por lo que no necesita interactuar con los servicios web de Amazon.

I.3.1. Parámetros de entrada

A continuación se pasan a describir los parámetros de entrada que necesita el programa:

- *si_type*: tipo de instancia. Hay que seguir la nomenclatura empleada para la base de datos (muy importante tenerlo en cuenta).
- *system*: sistema operativo
- *cost*: dinero total que podemos gastar.
- *init_year*: año de la fecha de inicio.
- *init_month*: mes de la fecha de inicio.
- *init_day*: día de la fecha de inicio.
- *init_hour*: hora de la fecha de inicio.
- *init_minute*: minuto de la fecha de inicio.
- *init_second*: segundo de la fecha de inicio.
- *deadline*: horas disponibles para poder completar la tarea.
- *task_time*: tiempo que dura la tarea (en horas)
- *subzone*: zona específica donde queremos simular (importante que se encuentre en la zona del tipo de instancia que queremos)

Los parámetros son de libre elección (siempre y cuando sean consecuentes con el contenido de la base de datos) aunque para que el uso del simulador tenga sentido habría que considerar los datos introducidos en la aplicación principal (la de predicción) y la salida que nos proporciona.

El programa toma los parámetros de entrada de un fichero XML. La ruta del fichero debe pasarse al programa como primer parámetro de entrada. El fichero debe tener la siguiente estructura:

```
<?xml version="1.0"?>
<input>
        <si_type>c1_medium_US</si_type>
        <system>Linux/UNIX</system>
        <cost>10 </cost>
        <init_year>2014</init_year>
        <init_month>6</init_month>
        <init_day>30</init_day>
        <init_hour>10</init_hour>
        <init_minute>00</init_minute>
        <init_second>00</init_second>
        <deadline>24</deadline>
        <task_time>5</task_time>
        <subzone>us-west-2c</subzone>
</input>
```

Tabla I.5: Estructura del fichero XML de entrada

I.3.2. Ejecución

El programa toma los parámetros necesarios del fichero XML. La ruta del fichero debe pasarse al programa como primer parámetro. Una vez recogida la información de entrada comienza su ejecución.

La fecha que se le facilita a través de estos parámetros es la fecha a partir de la cual se empezará la simulación. No es necesario que la fecha sea exacta, el simulador cogerá la más próxima y se ejecutará con normalidad.

El programa funciona como si de una puja real se tratase, mira el histórico y en caso de superar la puja se le asignan los recursos al usuario (siempre que esté dentro de su *deadline*).

Una vez se han asignado los recursos sólo hay dos escenarios posibles, que la tarea se complete o que no lo haga. Las horas de la tarea deben ser consecutivas y en caso de perder en un solo instante los recursos se considera fallida la simulación.

En la figura I.7 puede verse la salida del programa:

```
Task info:
---------------------------------
Spot Instance: c1_medium_AP
Cost (money): 0.21
Init date: 2015-07-01 01:00:00.217
Task time (hours): 10
Zone: ap-northeast-1c

PUJA: 0,0210

FECHA INICIO SOLICITADA: 2015-07-01 01:00:00.217
FECHA INICIO TAREA: 2015-07-01 01:00:00.217
FECHA FIN TAREA: 2015-07-01 11:00:00.217
RESULTADO;COSTE TOTAL: EXITO;0,2031
```

Figura I.7: Salida del programa de simulación

ANEXO II: Manual de usuario

II.1. Base de datos

En este apartado se detallan los pasos que se siguieron para instalar la base de datos.

- Es necesario tener cualquier sistema operativo compatible con MySQL 5.6.
- Posteriormente hay que crear una base de datos dentro del gestor MySQL disponible.
- A continuación se tienen que crear las tablas de la base de datos que almacenarán la información. Para ello es recomendable crear un *script* que genere todas las tablas (los dos tipos de los que se habló en la memoria). En la tabla II.1 puede verse un ejemplo de código.

```
CREATE TABLE IF NOT EXISTS zone_updates (

        Zone            VARCHAR(20) NOT NULL,
        Time            TIMESTAMP NOT NULL,
        PRIMARY KEY (Zone)
);

INSERT INTO zone_updates (Zone,Time)
VALUES ("AP_NORTHEAST_1",'2015-08-15 11:22:03');

INSERT INTO zone_updates (Zone,Time)
VALUES ("AP_SOUTHEAST_1",'2015-08-15 11:22:49');

INSERT INTO zone_updates (Zone,Time)
VALUES ("AP_SOUTHEAST_2",'2015-08-15 11:22:43');

INSERT INTO zone_updates (Zone,Time)
VALUES ("EU_CENTRAL_1",'2015-08-15 11:08:22');

INSERT INTO zone_updates (Zone,Time)
VALUES ("EU_WEST_1",'2015-08-15 10:57:21');

INSERT INTO zone_updates (Zone,Time)
VALUES ("SA_EAST_1",'2015-08-15 15:49:35');

INSERT INTO zone_updates (Zone,Time)
VALUES ("US_EAST_1",'2015-08-15 11:15:40');
```

```
INSERT INTO zone_updates (Zone,Time)
VALUES ("US_WEST_1",'2015-08-15 11:17:21');

INSERT INTO zone_updates (Zone,Time)
VALUES ("US_WEST_2",'2015-08-15 11:18:04');
```

Tabla II.1: Código para generar la tabla de zonas

```
CREATE TABLE IF NOT EXISTS c1_medium_AP (
        id                  BIGINT NOT NULL AUTO_INCREMENT,
        Zone                VARCHAR(20) NOT NULL,
        Description         VARCHAR(40) NOT NULL,
        Price               FLOAT NOT NULL,
        Time                TIMESTAMP NOT NULL,
        WeekDay             VARCHAR(5) NOT NULL,
        TimeZone            VARCHAR(5) NOT NULL,
        PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS c1_xlarge_AP (
        id                  BIGINT NOT NULL AUTO_INCREMENT,
        Zone                VARCHAR(20) NOT NULL,
        Description         VARCHAR(40) NOT NULL,
        Price               FLOAT NOT NULL,
        Time                TIMESTAMP NOT NULL,
        WeekDay             VARCHAR(5) NOT NULL,
        TimeZone            VARCHAR(5) NOT NULL,
        PRIMARY KEY (id)
);
```

Tabla II.2: Ejemplo de código para generar las tablas

II.2. Script AWK procesado de históricos

El primero de los *scripts* es el encargado de procesar los históricos y generar sentencias SQL de inserción para la base de datos. Se ejecuta de la siguiente manera:

awk -v bd=tfg -f generator.awk AP_NORTHEAST_1.txt

En este caso “tfg” es el nombre de la base de datos, sustituir por el que queramos, “generator.awk” es el propio nombre del script y “AP_NORTHEAST_1.txt” es el fichero de históricos que tenemos generado por la aplicación de descarga de históricos (cambiar por el fichero que queramos de históricos).

II.3. Aplicaciones Java

Para el desarrollo y ejecución se utilizó el IDE Eclipse [W5], concretamente la versión 3.8 usando Java 7.

Además, hay que integrar el SDK de Amazon en el IDE (Eclipse), hay disponible un tutorial en su web [W3].

Las 3 aplicaciones Java necesitan la librería "mysql-connector-java-5.1.33-bin" [W6]. La aplicación principal (cálculo de viabilidad) requiere además del uso de jfreechart-1.0.19 [W4], un conjunto de librerías para generar gráficas.

Para la ejecución del predictor y del simulador pueden encontrarse más detalles en el anexo I de esta memoria.

II.4. Gnuplot

Instalación de gnuplot mediante línea de comandos:

```
$ sudo apt-get install gnuplot
$ sudo apt-get install gnuplot-x11
```

Para ejecutar el script que aparece en el anexo correspondiente a gnuplot:

awk -v Type_I=c3.8xlarge -v Day1=01 -v Month1=07 -v Year1=2014 -v Day2=31 -v Month2=07 -v Year2=2014 -v SI_Zone=us-west-2a -v SO=Linux/UNIX -f generator2.awk log2.txt

Type_I: Tipo de instancia.
Day1: Día de la fecha 1.
Month1: Mes de la fecha 1.
Year1: Año de la fecha 1.
Day2: Día de la fecha 2.
Month2: Mes de la fecha 2.
Year2: Año de la fecha 2.
SI_Zone: subzona exacta donde está la máquina.
SO: Sistema operative.
log2.txt es el fichero con la información, sustituir por el que vayamos a usar.

Puede encontrarse más información del uso de Gnuplot en el Anexo III de esta memoria.

ANEXO III: Generación de gráficas mediante GNUPLOT

III.1. Obtención de los ficheros de información necesarios mediante AWK

Para la creación de ficheros que posteriormente uso en GNUPLOT para la generación de gráficas me apoyo en un script en AWK. A continuación puede verse el código del script:

```
#! /bin/awk -f
# Programa awk que transforma el fichero de logs en sentencias sql
# de inserción de datos para la base de datos pasada por parámetros.
# Ejemplo de ejecución: awk -v Type_I=c3.8xlarge -v Day1=01 -v Month1=07 -v Year1=2014 -v Day2=31 -v Month2=07 -v Year2=2014 -v SI_Zone=us-west-2a -v SO=Linux/UNIX -f generator2.awk log2.txt

BEGIN {
	FS = "[:|,|{|}]"
}

{
	#Variables de entrada

	#Asigno variables obtenidas del fichero y elimino espacios con "sub"
	Type=$3
	desc=$5
	sub(" ","",desc);
	price=$7
	sub(" ","",price);
	Zone=$13
	sub(" ","",Zone);
	weekday=substr($9,2,3);
	Year=substr($11,8,5);
	Month=substr($9,6,3);
	Day=substr($9,10,2);
	hour=substr($9,13,2);
	minute=$10
	gsub(" ","",minute);
	second=substr($11,1,2);
	timezone=substr($11,4,4);

	#Transformar mes
	if (Month == "Jan") {
		Month=01;
```

```
        }
        else if (Month == "Feb") {
                Month=02;
        }
        else if (Month == "Mar") {
                Month=03;
        }
        else if (Month == "Apr") {
                Month=04;
        }
        else if (Month == "May") {
                Month=05;
        }
        else if (Month == "Jun") {
                Month=06;
        }
        else if (Month == "Jul") {
                Month=07;
        }
        else if (Month == "Aug") {
                Month=08;
        }
        else if (Month == "Sep") {
                Month=09;
        }
        else if (Month == "Oct") {
                Month=10;
        }
        else if (Month == "Nov") {
                Month=11;
        }
        else if (Month == "Dec") {
                Month=12;
        }

        #Modifico el nombre de la tabla para cambiar el "." por un "_" y quitar un
espacio
        sub(" ","",Type);

        sub(" ","",year);
         sub(",","",year);
```

```
        sub(" ","",timezone);

        #Construyo la fecha
        date=Year"/"Month"/"Day"-"hour":"minute":"second

        #print "THIS"Type_I" "Day1" "Month1" "Year1" "Day2" "Month2" "Year2"
"SI_Zone" "SO" THIS!!!"

        # Check date - Si es del TIPO y mayor que dia mes y año 1 y menor que dia mes
y año 2, meter el precio en el fichero
        Correcto1 = "FALSE"
        Correcto2 = "FALSE"

        if ( SI_Zone == Zone && SO == desc ){
                if ( Type == Type_I ){
                        #Comprobación primera fecha
                        if (Year > Year1){
                                Correcto1 = "TRUE"
                        }
                        else if (Year == Year1){
                                if (Month > Month1){
                                        Correcto1="TRUE"
                                }
                                else if (Month == Month1){
                                        if (Day >= Day1){
                                                Correcto1="TRUE"
                                        }
                                        else{
                                                Correcto1="FALSE"
                                        }
                                }
                                else{
                                        Correcto1="FALSE"
                                }
                        }
                        else{
                                Correcto1="FALSE"
                        }

                        #Comprobación segunda fecha
```

```
                        if (Year < Year2){
                                Correcto2="TRUE"
                        }
                        else if (Year == Year2){
                                if (Month < Month2){
                                        print "OK"
                                }
                                else if (Month == Month2){
                                        if (Day <= Day2){
                                                print "OK"
                                        }
                                        else{
                                                Correcto2="FALSE"
                                                print "MAL"
                                        }
                                }
                                else{
                                        Correcto2="FALSE"
                                        print "MAL"
                                }
                        }
                        else{
                                Correcto2="FALSE"
                                print "MAL"
                        }
                }

                # Meter al fichero si el dato es correcto
                if (Correcto1 == "TRUE" && Correcto2 == "TRUE"){
                        print date" "price >> "gnu.dat"
                }
        }
}
```

Tabla III.1: Código AWK del script de generación de datos para GNUPLOT

III.2. Comandos utilizados para generar las gráficas

En este apartado se describen algunos de los comandos utilizados.

- *set title*: fija el título de la gráfica.
- *set xlabel*: modifica el nombre del eje X.
- *set ylabel*: modifica el nombre del eje Y.

- *set xdata time*: indica al programa que los datos del eje X son de carácter temporal.
- *set timefmt*: establece el formato de los datos de tipo fecha.
- *set format x*: fija cómo se va a visualizar la información del eje X.
- *set xrange*: establece el rango de visualización del eje X.
- *set yrange*: establece el rango de visualización del eje Y.
- *set term*: fija el formato de salida de los resultados.
- *set output*: nombre del fichero de salida que recoge los resultados.
- *plot*: dibujar el fichero pasado por parámetros.

III.3. Ejemplo de uso

En la tabla pueden verse los comandos anteriormente descritos en un ejemplo práctico.

```
gnuplot> set title "Price variation in 24 hours\nc1_medium instances (US)"
gnuplot> set xlabel "Hour"
gnuplot> set ylabel "Price"
gnuplot> set xdata time
gnuplot> set timefmt "%d/%m/%Y-%H:%M:%S"
gnuplot> set format x "%H:%M"
gnuplot> set xrange ["30/07/2014-00:00:00":"31/07/2014-00:00:00"]
gnuplot> set yrange [0.015:0.018]
gnuplot> set term png
gnuplot> set output "output.png"
gnuplot> plot "gnu.dat" using 1:2 with lines
```

Tabla III.2: Ejemplo de comandos para la generación de gráficas con GNUPLOT

A continuación se muestra la figura III.1 [W9] con algunas aclaraciones sobre los comandos y su uso en el ejemplo anterior.

Time Series timedata Format Specifiers	
Format	Explanation
%d	day of the month, 1-31
%m	month of the year, 1-12
%y	year, 0-99
%Y	year, 4-digit
%j	day of the year, 1-365
%H	hour, 0-24
%M	minute, 0-60
%s	seconds since the Unix epoch (1970-01-01 00:00 UTC)
%S	second, 0-60
%b	three-character abbreviation of the name of the month
%B	name of the month

Figura III.1: Leyenda del formato de GNUPLOT

ANEXO IV: Gestión del proyecto

En este anexo se detalla la metodología seguida durante el proyecto así como las fases en las que se ha dividido y cuánto tiempo ha llevado cada una de ellas. También se explica la supervisión llevada por el director y las herramientas de gestión utilizadas.

IV.1. Metodología de desarrollo

Durante el proyecto se ha seguido una metodología de desarrollo con ciclo de vida en cascada teniendo retroalimentación en cada una de las fases. En la figura IV.1 pueden verse cada una de las fases de las que consta el ciclo y cómo la retroalimentación obtenida de cualquiera de ellas puede afectar a las anteriores.

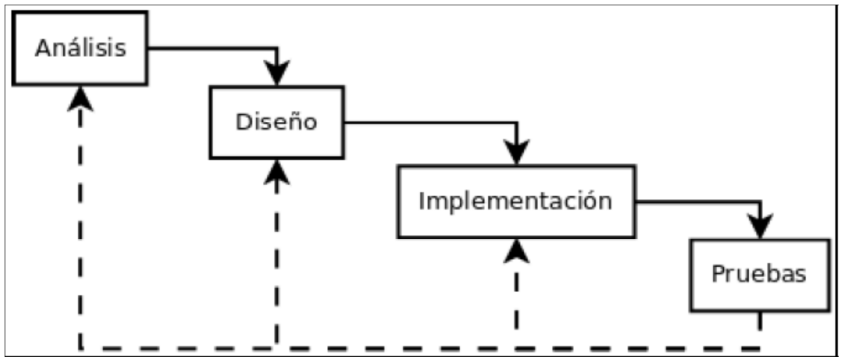


Figura IV.1: Ciclo de vida en cascada con retroalimentación.

IV.2. Fases del proyecto

En este apartado se explican de forma breve las diferentes fases que se han seguido en el proyecto así como un listado de las tareas más relevantes que se llevaron a cabo durante las mismas.

1. Puesta en contexto. En esta primera fase se estudió el contexto en el que se desarrolla el proyecto. Se buscó información sobre las instancias puntuales y publicaciones donde trataban los puntos clave de su funcionamiento, así como las herramientas que nos ofrecía Amazon. Se estudiaron los siguientes conceptos:
   - Computación *cloud.*
   - Instancias puntuales.
   - Recursos para desarrolladores de Amazon EC2.

2. Análisis. En esta fase se llevó a cabo un análisis del sistema que se quería realizar basado en la información obtenida en el primero punto. Se establecieron los requisitos del sistema. Las tareas principales fueron:
   - Análisis del funcionamiento de las instancias puntuales.
   - Análisis del modo de evaluación de viabilidad de una tarea.

- Definición de los requisitos del sistema así como de las tareas a ejecutar.

3. Diseño de la solución. En esta fase se define la arquitectura del sistema así como las tecnologías utilizadas en cada componente. Las tareas fundamentales desarrolladas fueron:
   - Diseño de la base de datos.
   - Diseño de la aplicación principal (predictor).
   - Diseño del simulador.

4. Implementación. Esta fase corresponde con todas las tareas relacionadas con el traslado del diseño a la tecnología elegida en cada caso. Los sistemas que se desarrollaron son los siguientes:
   - Desarrollo del obtenedor de histórico de precios.
   - Desarrollo del predictor.
   - Desarrollo del simulador.
   - Scripts de procesamiento de históricos.
   - Puesta en marcha de la base de datos.
   - Desarrollo y configuración de la automatización de la base de datos.

5. Pruebas. Incluye todas las tareas relacionadas con la elaboración y ejecución de tareas para el análisis del sistema implementado. El objetivo es valorar la efectividad del sistema así como extraer nuevas conclusiones de éste.
   - Pruebas con el predictor.
   - Pruebas con el simulador.

6. Documentación. Esta fase se llevó a cabo durante todo el proyecto. Engloba la elaboración de todos los documentos realizados durante el proyecto, ya sea incluida o no en la memoria. Podría dividirse en 2 grupos:
   - Documentación de la memoria.
   - Documentación de soporte para la realización del trabajo.

IV.3. Gestión del tiempo y esfuerzo

En esta sección se detalla el tiempo dedicado a cada una de las fases del proyecto así como el esfuerzo dedicado a las mismas.

IV.3.1. Gestión del tiempo

En la figura IV.2 se puede apreciar el diagrama de Gantt con la distribución de tiempo dedicada a cada una de las fases expuestas en el anterior apartado.

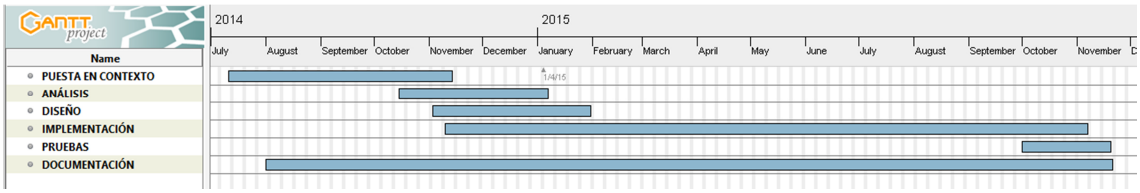


Figura IV.2: Diagrama de Gantt con distribución de fases del proyecto.

El proyecto empezó tras realizar una reunión inicial al finalizar el cuatrimestre. En ese momento se me facilitó una bibliografía con la que comenzar a conocer el contexto del proyecto así como el trabajo ya realizado por otras personas en este campo.

Se observa una etapa de puesta en contexto más duradera de lo que cabría esperar. El motivo fue la preparación de la última asignatura a superar del grado, por lo que se relajó esta fase hasta la finalización de los exámenes.

Cuando se estaba finalizando la puesta en contexto, se comenzó a analizar qué requisitos debía cumplir el sistema y sobretodo definir unos objetivos concretos en los que focalizar el análisis. Se partía de las publicaciones que intentaban establecer un modelo de variación de precios que se ajustase al de Amazon. Ante la complejidad de las publicaciones, del sistema de Amazon y de las innumerables posibilidades que brindaba un proyecto de esta tipología hubo que definir unas limitaciones con el fin de no desviarnos de lo esencial.

A pesar de que no todos los objetivos estuvieron definidos desde el comienzo de la fase de diseño se comenzó a trabajar en aquellos que estaban claros y que iban a ser estrictamente necesarios, como por ejemplo el uso de una base de datos y la creación de una aplicación que obtuviese los precios a través de los servicios web de Amazon. Es por ello que las fases aparecen solapadas. Se establecieron qué tecnologías se usarían para abordar la futura implementación.

Del mismo modo que sucedía entre el análisis y el diseño, en la fase de implementación también existe una solapación con las otras 2 fases. No hubo problema ya que las primeras implementaciones (BD y obtenedor de precios) eran componentes independientes del resto del futuro sistema. Por lo tanto mientras se analizaba y diseñaba el resto de elementos que compondrían el sistema se pudo comenzar con la implementación.

En febrero se observa un parón importante. A principios de ese mes comencé a trabajar en una empresa y el proyecto quedó relegado a un segundo plano durante varios meses. Se retomó en septiembre.

En la última etapa del proyecto, se acabó de implementar el sistema y se comenzó con la fase de pruebas que concentra toda su actividad en octubre y noviembre. La fase de

pruebas tuvo que realizarse en paralelo con parte de la implementación al tener que preparar diversas baterías de pruebas así como la consideración de nuevas casuísticas que no se habían contemplado hasta la fecha. En estas fechas se finaliza la documentación a pesar de que es una fase en la que se ha ido trabajando a lo largo de todo el proyecto.

IV.3.2. Gestión del esfuerzo

En la figura IV.3 se puede ver el porcentaje de esfuerzo dedicado a cada fase del proyecto.

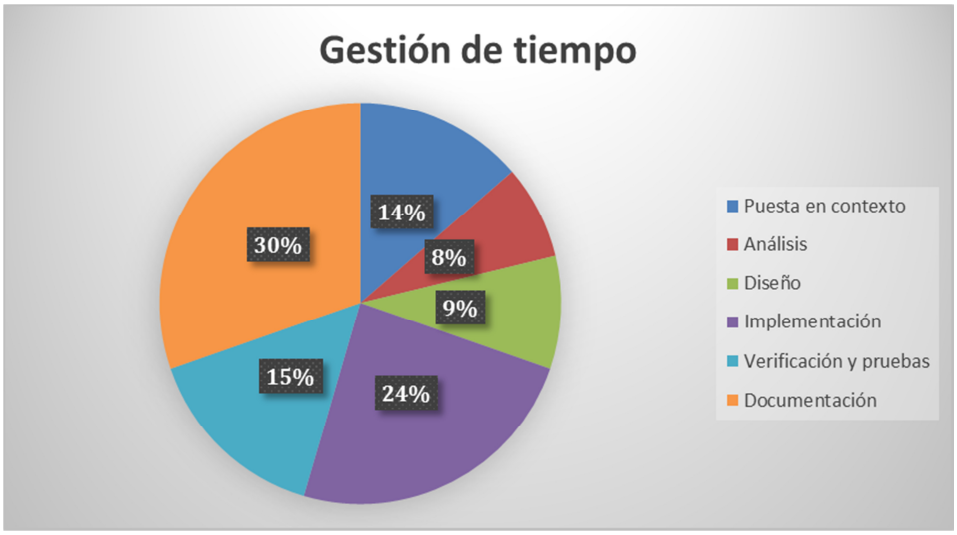


Figura IV.3: Distribución del porcentaje de tiempo por fase.

El total de horas dedicadas al proyecto asciende a 330 horas. A continuación se puede observar el número de horas dedicado a cada una de las fases:

- Puesta en contexto: 45 horas
- Análisis: 25 horas
- Diseño: 30 horas
- Implementación: 80 horas
- Verificación y pruebas: 50 horas
- Documentación: 100 horas

IV.4. Supervisión del proyecto

Durante la realización del proyecto se ha contado con la supervisión de un director y un ponente con los cuales se han ido haciendo reuniones para controlar los avances del proyecto así como para definir la dirección del mismo en cada hito.

No hubo una periodicidad definida para las reuniones, se convocaron conforme surgía la necesidad de plantear un cambio en el proyecto o simplemente para comprobar el estado del mismo.

Por norma general, las reuniones seguían la siguiente estructura:

1. Orden del día.
2. Estado de situación del proyecto
3. Evaluación del trabajo realizado.
4. Discusión y debate de ideas, dudas e impresiones.
5. Planificación de nuevos objetivos.
6. Planificación de la siguiente reunión.

Durante la reunión se elaboraba un acta con los aspectos relevantes tratados. Estas actas han sido de gran ayuda tanto para realizar el seguimiento del proyecto como para consultar las decisiones tomadas a lo largo del mismo.

Además de las reuniones, hubo seguimiento por parte de los directores siempre y cuando fue requerido.

IV.5. Herramientas de gestión utilizadas

Para la gestión del proyecto se usaron diversas herramientas para facilitar su realización. Dichas herramientas afectan a la gestión documental, la seguridad y la comunicación entre el alumno y los directores.

En primer lugar, se utilizó Dropbox [W19] para compartir y mantener la documentación del proyecto. Dropbox es un sistema que nos permite almacenar y compartir archivos a través de Internet. Es una herramienta sencilla, gratuita y que además incluye un control de versiones por lo que siempre se puede recuperar un trabajo anterior. Además, se utilizó para guardar copias periódicas de los sistemas desarrollados en caso de necesitar una copia de seguridad. Dichas copias se almacenaban también en el equipo local y en memorias externas.

Para la comunicación de los directores con el alumno se usó el correo electrónico y Skype [W20]. Con el primero de ellos. Estas herramientas permitieron la comunicación entre los implicados en el proyecto de forma rápida y eficaz. La utilización de Skype permitió otro grado de interacción facilitando la realización de reuniones cuando no era posible hacerlo de modo presencial.

Bibliografía

[B1] Javadi, Bahman, Ruppa K. Thulasiram, and Rajkumar Buyya. "Characterizing spot price dynamics in public cloud environments." *Future Generation Computer Systems* 29.4 (2013): 988-999.

[B2] Agmon Ben-Yehuda, Orna, et al. "Deconstructing Amazon EC2 spot instance pricing." *ACM Transactions on Economics and Computation* 1.3 (2013): 16.

[B3] Wee, Sewook. "Debunking real-time pricing in cloud computing." *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*. IEEE, 2011.

[B4] Song, Yang, Murtaza Zafer, and Kang-Won Lee. "Optimal bidding in spot instance market." *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012.

[B5] Zafer, Murtaza, Yang Song, and Kang-Won Lee. "Optimal bids for spot VMs in a cloud for deadline constrained jobs." *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 2012.

[B6] Javadi, Bahman, Ruppa K. Thulasiram, and Rajkumar Buyya. "Statistical modeling of spot instance prices in public cloud environments." *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*. IEEE, 2011.

[B7] Urgaonkar, Bhuvan, et al. "Pricing of service in clouds: optimal response and strategic interactions." *ACM SIGMETRICS Performance Evaluation Review* 41.3 (2014): 28-30.

[B8] Zhang, Qi, Quanyan Zhu, and Raouf Boutaba. "Dynamic resource allocation for spot markets in cloud computing environments." *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*. IEEE, 2011.

[B9] Chaisiri, Sivadon, et al. "Cost minimization for provisioning virtual servers in amazon elastic compute cloud." *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on*. IEEE, 2011.

[B10] Taylor, Ian J., et al. Workflows for e-Science: scientific workflows for grids. Springer Publishing Company, Incorporated, 2014.

[B11] Smolan, Rick. "The human face of big data." (2013).

[B12] Foster, Ian, and Carl Kesselman, eds. *The Grid 2: Blueprint for a new computing infrastructure*. Elsevier, 2003.

[B13] Berman, Fran, Geoffrey Fox, and Anthony JG Hey. *Grid computing: making the global infrastructure a reality*. Vol. 2. John Wiley and sons, 2003.

[B14] Opitz, Alek, Hartmut König, and Sebastian Szamlewska. "What does grid computing cost?." Journal of Grid Computing 6.4 (2008): 385-397.

[B15] Risch, Marcel, and Jörn Altmann. "Cost analysis of current grids and its implications for future grid markets." Grid Economics and Business Models. Springer Berlin Heidelberg, 2008. 13-27.

[B16] Foster, Ian, et al. "Cloud computing and grid computing 360-degree compared."Grid Computing Environments Workshop, 2008. GCE'08. Ieee, 2008.

[B17] Armbrust, Michael, et al. "A view of cloud computing." Communications of the ACM 53.4 (2010): 50-58.

[B18] Rehr, John J., et al. "Scientific computing in the cloud." Computing in science & Engineering 12.3 (2010): 34-43.

[B19] Wang, Lei, et al. "In cloud, can scientific communities benefit from the economies of scale?." Parallel and Distributed Systems, IEEE Transactions on23.2 (2012): 296-303.

[B20] Carlyle, Adam G., Stephen L. Harrell, and Preston M. Smith. "Cost-effective HPC: The community or the Cloud?." Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. IEEE, 2010.

[B21] De Alfonso, Carlos, et al. "An economic and energy-aware analysis of the viability of outsourcing cluster computing to a cloud." Future Generation Computer Systems 29.3 (2013): 704-712.

[B22] He, Qiming, et al. "Case study for running HPC applications in public clouds."Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. ACM, 2010.

[B23] Gupta, Abhishek, and Dejan Milojicic. "Evaluation of hpc applications on cloud."Open Cirrus Summit (OCS), 2011 Sixth. IEEE, 2011.

[W1] Amazon Virtual Private Cloud
http://aws.amazon.com/es/vpc/

[W2] Instancias reservadas de Amazon EC2
https://aws.amazon.com/es/ec2/purchasing-options/reserved-instances/

[W3] AWS Toolkit for Eclipse
http://aws.amazon.com/es/eclipse/

[W4] FreeChart, Java chart library
http://www.jfree.org/jfreechart/

[W5] Java IDE Eclipse
https://eclipse.org/

[W6] JDBC driver for MySQL
http://dev.mysql.com/downloads/connector/j/

[W7] Spot Instance Pricing History
http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances-history.html

[W8] Amazon Web Services
https://aws.amazon.com/es/what-is-aws/

[W9] Documentación Gnuplot
http://gnuplot.sourceforge.net/docs_4.2/node274.html

[W10] Amazon Elastic Compute Cloud
https://aws.amazon.com/es/ec2/

[W11] Amazon EC2 spot request volatility hits $1000/hour
https://moz.com/devblog/amazon-ec2-spot-request-volatility-hits-1000hour/

[W12] Instancias puntuales de Amazon EC2
https://aws.amazon.com/es/ec2/spot/

[W13] Tamaño de la muestra
https://es.wikipedia.org/wiki/Tama%C3%B1o_de_la_muestra

[W14] Microsoft Azure
https://azure.microsoft.com/en-us/

[W15] Google Cloud Platform
https://cloud.google.com/appengine/

[W16] Office
https://www.office.com/

[W17] Google Cloud Platform
https://cloud.google.com/compute/

[W18] Google Docs
https://www.google.es/intl/es/docs/about/

[W19] Dropbox
https://www.dropbox.com/es/

[W20] Skype
http://www.skype.com/es/