Informe de Elasticidad

Marilin Vega León
Iván Pajares Fuente

El siguiente es informe tiene como objetivo la realización de una práctica que sigue el modelo dado en el capítulo 8 del libro “Performance by Design: Computer Capacity Planning by Example de Daniel A. Menascé”, para construir un controlador de elasticidad que permita, a partir de una configuración, encontrar una mejor configuración para soportar la carga de operaciones de entrada al sistema.

Actividades:

1º. Construir el modelo de la aplicación. A partir del empleo de los algoritmos MVA multiclase construir la gráfica que representa (para una configuración dada) el tiempo de respuesta frente a la carga de entrada global del sistema. Realizar distintas gráficas con diferentes configuraciones y comparar los resultados.

2º Definir en base a los resultados obtenidos sobre el sistema los niveles de QoS que servirán como base para evaluar diferentes configuraciones. Básicamente se trata de definir cuál es el tiempo de respuesta objetivo, bien para cada clase de operaciones o un tiempo global medio. Para evaluar una configuración con una carga dada debéis proponer alguna función de utilidad que indique la relación del coste (número de instancias en cada capa para una configuración), el tiempo de respuesta, y la carga soportada.

3º Idear un proceso de elasticidad que se base en las medidas que es razonable extraer del sistema en funcionamiento.

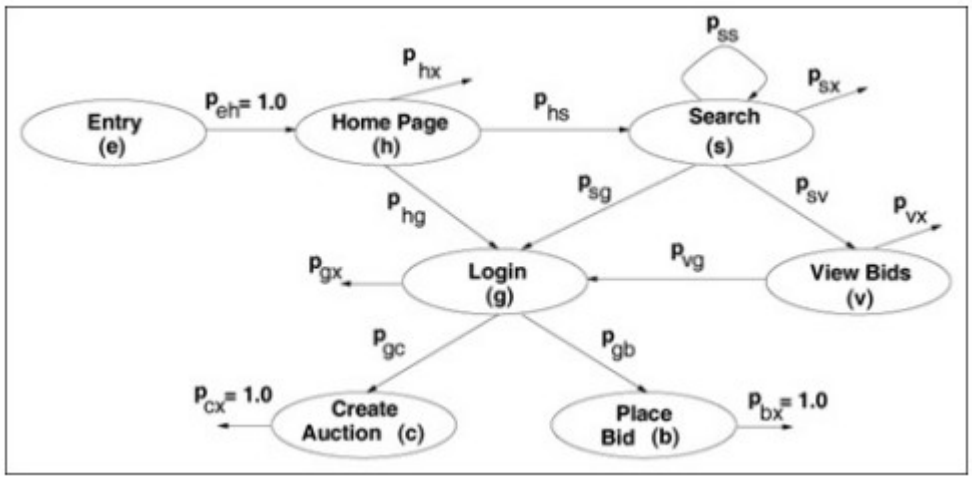
Concepto de Elasticidad (según Herbst)

Es el grado por el cual un sistema es capaz de adaptarse a los cambios en la carga de trabajo que soporta mediante el abastecimiento y desabastecimiento de recursos de cómputo de forma autónoma, de tal manera que en cada punto del tiempo los recursos disponibles encajan con la demanda de recursos real tan cerca como sea posible.

Un concepto asociado es el de escalabilidad, el cual brinda la base para que un sistema poder tener cierto grado de elasticidad. Este indica la habilidad de un sistema para reaccionar y adaptarse sin perder la calidad, también para manejar el crecimiento continuo de trabajo de manera fluida, o de una forma más sencilla estar preparado para crecer.

En el caso de la práctica nos enfocaremos en un escalado horizontal, y se lograra adicionando o eliminando componentes teniendo en cuenta en que cantidad, esto se consigue mediante un proceso de adaptación que valla indicando que decisión tomar a medida que valla cambiando.

Grafo a usar para la práctica:



Breve descripción:

-Este es un grafo que representa cualquier tipo de aplicación Web de manera general.

Nodo (e) -> representa al cliente entrando al sitio.

Arcos -> representan la probabilidad de transición directa de nodo a nodo.

Los nodos de salida (x) -> es el estado al que el cliente salir del sistema desde cualquier estado i con probabilidad Pix.

Procedimiento:

1 - Se obtienen las cargas de trabajo para cada tipo de clase de operación. Este proceso comienza obteniendo las visitas a cada estado realizadas por un cliente en una sesión. Se usan las tablas 1 y tabla 2 para obtener las probabilidades de transición de los estados.

Fórmulas:

$$V_e = 1$$
$$V_h = V_e \times p_{eh} = 1$$
$$V_s = V_h \times p_{hs} + V_s \times p_{ss}$$
$$V_v = V_s \times p_{sv}$$
$$V_g = V_h \times p_{hg} + V_s \times p_{sg} + V_v \times p_{vg}$$
$$V_c = V_g \times p_{gc}$$
$$V_b = V_g \times p_{gb}$$

*Fórmulas correspondientes al grafo anterior

	(e)	(h)	(s)	(v)	(g)	(c)	(b)	(x)
Entry (e)	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
Home (h)	0.00	0.00	0.70	0.00	0.10	0.00	0.00	0.20
Search (s)	0.00	0.00	0.40	0.20	0.15	0.00	0.00	0.25
View Bids (v)	0.00	0.00	0.00	0.00	0.65	0.00	0.00	0.35
Login (g)	0.00	0.00	0.00	0.00	0.00	0.30	0.60	0.10
Create Auction (c)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
Place Bid (b)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
Exit (x)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Tabla 1

	(e)	(h)	(s)	(v)	(g)	(c)	(b)	(x)
Entry (e)	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
Home (h)	0.00	0.00	0.70	0.00	0.10	0.00	0.00	0.20
Search (s)	0.00	0.00	0.45	0.15	0.10	0.00	0.00	0.30
View Bids (v)	0.00	0.00	0.00	0.00	0.40	0.00	0.00	0.60
Login (g)	0.00	0.00	0.00	0.00	0.00	0.30	0.55	0.15
Create Auction (c)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
Place Bid (b)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
Exit (x)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Tabla 2

2 – Una vez resuelto el sistema lineal anterior y obtenidos los resultados de las visitas, la definición de $\boldsymbol{\lambda}$ como la carga total que entra al sistema y teniendo a f_A y f_B

$\boldsymbol{\lambda} = 1$

$f_A = 0.25$

$f_B = 0.75$

Se procede a calcular la carga para cada una de las anteriores clases.

Formulas:

$$\lambda_h = \gamma(f_A \times V_h^A + f_B \times V_h^B)$$
$$\lambda_s = \gamma(f_A \times V_s^A + f_B \times V_s^B)$$
$$\lambda_v = \gamma(f_A \times V_v^A + f_B \times V_v^B)$$
$$\lambda_l = \gamma(f_A \times V_g^A + f_B \times V_g^B)$$
$$\lambda_c = \gamma(f_A \times V_c^A + f_B \times V_c^B)$$
$$\lambda_b = \gamma(f_A \times V_b^A + f_B \times V_b^B)$$

3 – Se procede a calcular la utilidad para definir cuan útil es el modelo.

Se hace uso de la matriz de demandas dada en la Tabla#3

Device	(h)	(s)	(v)	(g)	(c)	(b)
WS-CPU	0.008	0.009	0.011	0.060	0.012	0.015
WS-disk	0.030	0.010	0.010	0.010	0.010	0.010
AS-CPU	0.000	0.030	0.035	0.025	0.045	0.040
AS-disk	0.000	0.008	0.080	0.009	0.011	0.012
DS-CPU	0.000	0.010	0.009	0.015	0.070	0.045
DS-disk	0.000	0.035	0.018	0.050	0.080	0.090

Tabla #3

Formula:

$$U_i = \sum_{r=1}^{R} (\lambda_r / N_{\text{ws}})\ D_{i,r}$$

***para calcular las utilidades**

$$R_r = \sum_{i=1}^{K} \frac{D_{i,r}}{1 - U_i} = \sum_{i=1}^{K} \frac{D_{i,r}}{1 - \sum_{r=1}^{R} (\lambda_r / N_{\text{ws}})\ D_{i,r}}$$

***para calcular los tiempos de respuesta**

Resultados ejercicios:

Ejercicio 1:

Obtención de las visitas: A
e 1.000000
h 1.000000
s 1.167
v 0.233
g 0.427
c 0.128
b 0.256

Obtención de las visitas: B
e 1.000
h 1.000
s 1.273
v 0.191
g 0.304
c 0.091
b 0.167

Obtención de las cargas
('h', 11.0)
('s', 13.71)
('v', 2.22)
('g', 3.68)
('c', 1.11)
('b', 2.10)
('x', 11.0)

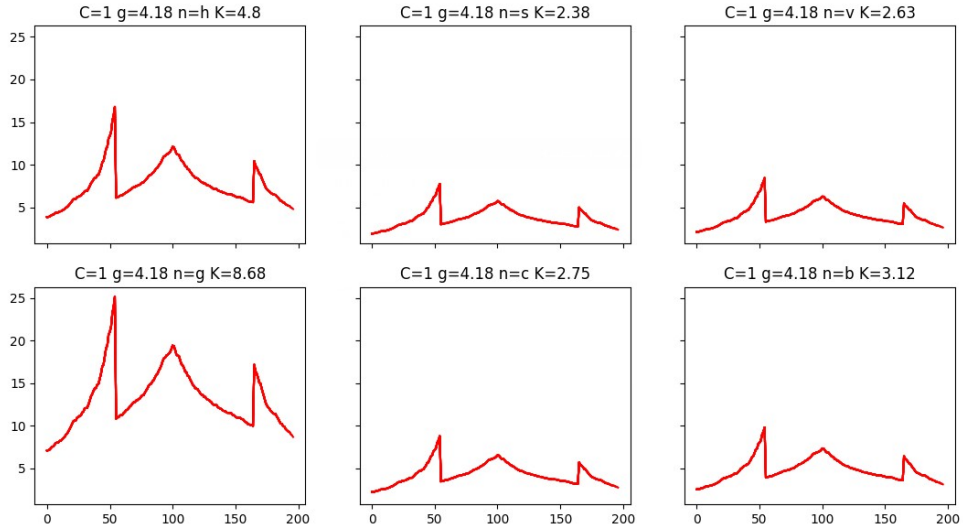
Obtención de las utilidades
('WS-CPU', 0.167)
('WS-disk', 0.186)

Obtención de los tiempos de respuesta
('h', 0.046)
('s', 0.023)
('v', 0.025)
('g', 0.084)
('c', 0.027)
('b', 0.030)

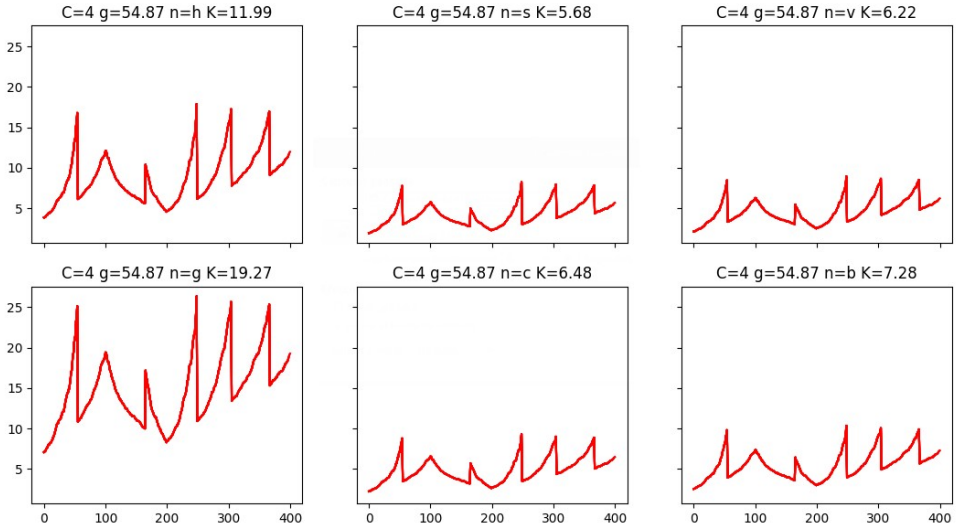
Vemos que los resultados obtenidos al programar el algoritmo explicado en el capítulo 8 del libro “Performance by Design: Computer Capacity Planning by Example de Daniel A. Menascé” coinciden con los resultados esperados, por lo que asumimos que la programación de las diferentes fórmulas es correcta y acertada.

Seguimos con la implementación de las funciones parametrizadas necesarias para la obtención de las gráficas de tiempos de respuesta para una tasa de entrada (gamma) y para un número de componentes por cada tipo.

Obtenemos las siguientes gráficas:



Iteraciones 0-200



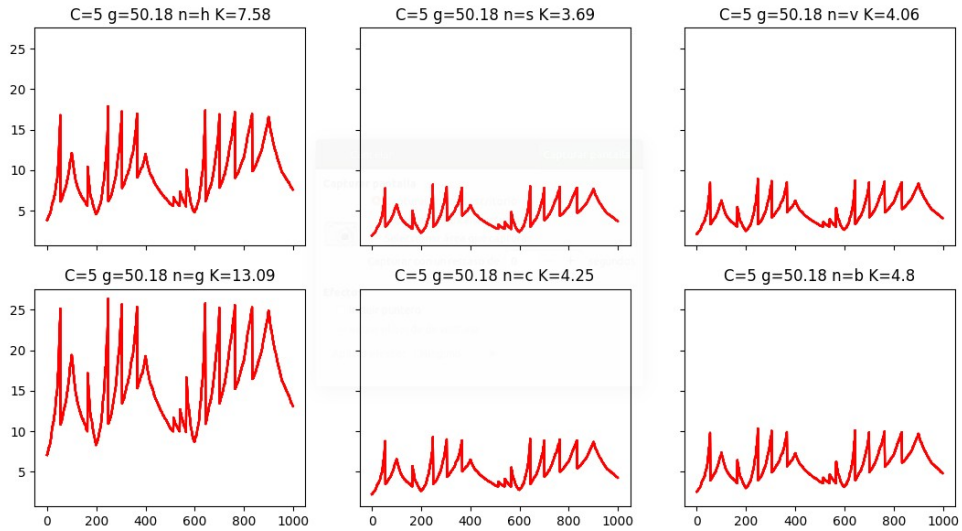
Iteraciones 0-400

C: número de componentes
g: gamma (tasa de entrada) actual
n: nodo (página de la web representada en el grafo)
K: tiempo de respuesta del nodo

Lo que hemos hecho ha sido:
- Generamos un gamma que va subiendo aleatoriamente entre 0 y 0,5 hasta que la iteración es 100, y baja con la misma probabilidad hasta la iteración 200; vuelve a subir hasta la 400, bajará hasta la 600, subirá hasta la 900,...
- Si el tiempo de respuesta (eje Y) supera 0,25 (25 en las gráficas) en alguna página (gráfica), añadimos un componente
- Si el tiempo de respuesta es menor que 0,1 (10 en las gráficas) en todas las páginas (gráficas), quitamos un componente

La progresión se explica de la siguiente manera:
- En la iteración 50, la página *g* supera el máximo tiempo de respuesta permitido, por lo que se añade un componente. Eso ocurre porque la *g (gamma)* se encuentra en el periodo de crecimiento. Ello hace que todas las páginas reduzcan su tiempo de respuesta
- En la iteración 100, la *g (gamma)* empieza a decrecer
- En la iteración 170, el tiempo de respuesta de todas las páginas se encuentra por debajo del mínimo exigible, por lo que se quita un componente. Eso ocurre porque la *g (gamma)* se encuentra en el periodo de decrecimiento. Ello hace que todas las páginas incrementen su tiempo de respuesta y lo sigan reduciendo hasta la iteración 200
Lo mismo ocurre en el resto de pares de periodos: 200-400 de crecimiento, 400-600 de decrecimiento,…

En la siguiente gráfica vemos la progresión hasta 1000 iteraciones.

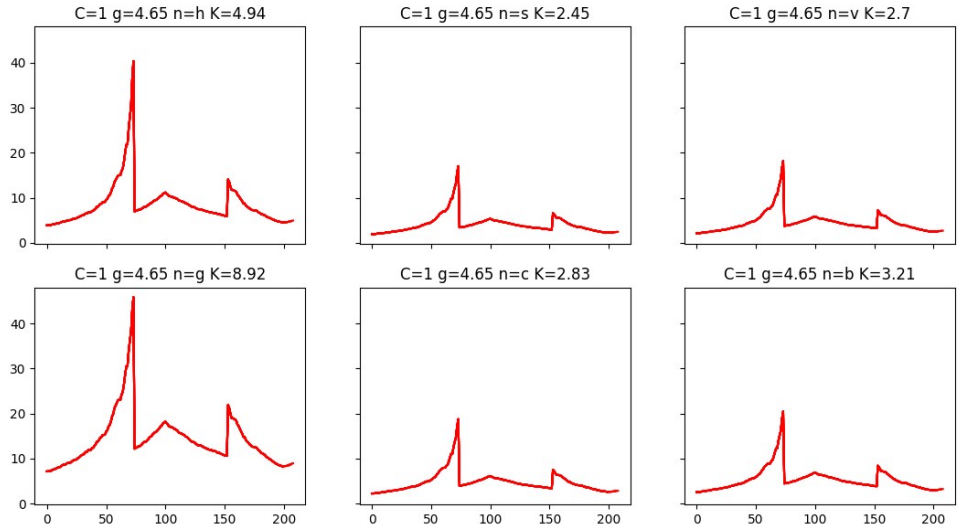


Ejercicio 2:

Queremos mantener un tiempo de respuesta inferior a 0,25 en cualquiera de las páginas para mantener la satisfacción de los usuarios en todas las interacciones con nuestro servicio. Ello es porque no queremos que ciertos usuarios que visitan más unas páginas que otros tengan ciertos beneficios respecto al resto. Por ello, en el apartado anterior hemos limitado el tiempo de respuesta a 0,25. Esto mismo podríamos hacerlo de diferente forma, como se explica a continuación.

- Tiempo medio de respuesta para todas las páginas:
Podemos limitar el tiempo medio de respuesta para que si, por ejemplo, una página tiene un alto tiempo de respuesta pero el resto no, no se añadan componentes para satisfacer la demanda de la que mayor tiempo de respuesta genera. Ello reduciría el tiempo que hay con un número mayor de componentes, pero penalizaría enormemente a los usuarios que visiten la página con mayor tiempo de respuesta.

La gráfica que resultaría con un tiempo medio máximo de 0,25 y un tiempo medio mínimo de 0,05 sería la siguientes



La diferencia con el ejercicio anterior respecto al tiempo mínimo reside en que si mantenemos el tiempo mínimo a 0,1, al añadir un componente, el tiempo medio baja demasiado y obliga a quitar componentes, lo que hace que el tiempo medio vuelva a subir y se formen *sierras* en la gráfica.

Además, la configuración de tiempo mínimo 0,05 para el ejercicio anterior imposibilita la reducción de componentes ya que el tiempo de respuesta de la gráfica de la página *g* nunca baja de 5 (0,05) y nunca estarían todas por debajo de el límite inferior para la eliminación de componentes.

Hemos visto dos enfoques diferentes con también diferencias en las configuraciones para hacer el sistema elástico. En los dos hemos configurado el sistema de forma que no se añadan y eliminen componentes en iteraciones cercanas en el tiempo para evitar *sierras* en las gráficas.

En el primero de ellos, lo que garantizamos es que todas las páginas tengan un tiempo de respuesta inferior a 0,25. Ello obliga a que, entre las iteraciones desde la 0 hasta la 200 haya aproximadamente 120 iteraciones con 2 componentes.

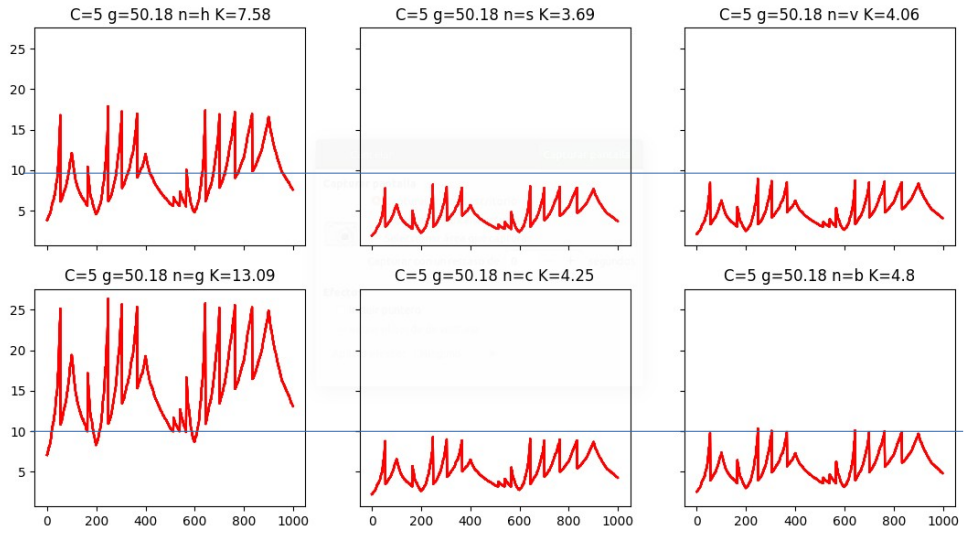
En el segundo, lo que garantizamos es que el servicio tenga un tiempo de respuesta medio inferior a 0,25. Ello obliga a que, entre las iteraciones desde la 0 hasta la 200 haya aproximadamente 75 iteraciones con 2 componentes.

En términos de rendimiento independientemente de la página que visiten los usuarios, el primer enfoque es mejor pero obliga a gastar más recursos, lo que incrementaría el coste del servicio. En el segundo enfoque, el rendimiento general es constante, beneficiando a los usuarios que más visiten las páginas con menor tiempo de respuesta, pero perjudicando a los que visitan las páginas *h* y *g* enormemente. Este segundo enfoque minimiza los recursos consumidos por el servicio, lo que económicamente hablando es más rentable.

Ejercicio 3

Proponemos un sistema de elasticidad basado en la reducción de la diferencia absoluta entre el tiempo que queremos garantizar para todas las páginas y el tiempo medio de respuesta obtenido para cada una de ellas, basándonos en el trabajo 3 que expusieron nuestros compañeros en clase: “Elasticity in Cloud Computing”.

Para explicar el sistema propuesto, vamos a trazar una línea en el tiempo de respuesta que queremos conseguir y sumar todas las áreas entre todas las gráficas y dicha línea. La suma de estas áreas será la función a minimizar.



En esta gráfica hemos pintado una línea azul en el tiempo de respuesta que queremos obtener para todas las páginas. El objetivo del sistema sería obtener en todas las páginas (gráficas) una gráfica lo más cercana posible a la línea azul para así minimizar el área comprendida entre la línea azul y las gráficas rojas.