Trabajo de Fin de Grado
Ingeniería de Organización Industrial

Desarrollo de un modelo de eventos discretos en Matlab para la operación óptima de Centros de Proceso de Datos.

Autor: Jaime Gil-Arévalo Gómez-Uribarri
Tutores: Daniel Limón Marruedo
Daniel Rodriguez Ramirez

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Trabajo de Fin de Grado
Ingeniería de Organización Industrial

Desarrollo de un modelo de eventos discretos en Matlab para la operación óptima de Centros de Proceso de Datos.

Autor:
Jaime Gil-Arévalo Gómez-Uribari

Tutor:
Daniel Limón Marruedo
Profesor Catedrático

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Proyecto Fin de Carrera: Desarrollo de un modelo de eventos discretos en Matlab para la operación óptima de Centros de Proceso de Datos.

Autor: Jaime Gil-Arévalo Gómez-Uribarri

Tutor: Daniel Limón Marruedo

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Quiero aprovechar esta oportunidad para agradecerles a mis padres todo el esfuerzo y la dedicación que han puesto en mí, para que lograrán llegar a donde voy llegando poco a poco.

Pero en esta ocasión quiero agradecerle especialmente a Daniel Limón, mi tutor, por todas las horas dedicadas, la gran predisposición a siempre echar una mano y por la gran pasión que le pone a su vocación, ya que así nos lo hace todo mucho mas ameno y sencillo a los demás

Tambien darle gracias Daniel Rodriguez, también mi tutor, por la disponibilidad prestada y la gran ayuda, incluso faltándole el tiempo casi para respirar.

Jaime Gil-Arévalo Gómez-Uribarri

Sevilla, 2019

Resumen

En este proyecto se va a tratar de modelar un Centro de Procesamiento de Datos (CPD) o Data Centers, para asi poder ver su comportamiento cuando este esta sometido a varias alteraciones externas manipulables o no.

La función de todo este proyecto es que a base de un modelado de eventos discretos consiga simularse con suficiente precisión el funcionamiento de un CPD, para así poder mejorar la eficiencia energética del mismo en un futuro.

Abstract

In this project we will try to model a Data Center so that we can see its behavior when it is exposed to several manipulable external alterations or not.

The function of this whole project is that, based on a modeling of discrete events, the operation of a Data Center can be simulated with enough precision, in order to improve its energy efficiency in the future.

Índice

ÍNDICE DE FIGURAS

1 ¿QUÉ ES UN CENTRO DE DATOS?

Los centros de datos son lugares dedicados a almacenar y procesar datos digitales. En sí, son grandes habitaciones o naves industriales repletas de servidores (ordenadores), agrupados a su vez en Clústers (racks), grandes estanterías metálicas, que favorecen a la refrigeración de dichos servidores al estar completamente abiertas y a la reducción de espacio.

Debido al gran incremento e importancia de los datos, algunas empresas se han visto obligadas a depender de dichos centros ya que el coste de tener un almacenamiento local físicamente dentro de la empresa se encarece.

1.1 Componentes de un centro de datos

Como ya he citado anteriormente, un centro de datos es una instalación que contiene hardware de tecnología de la información, como servidores, sistemas de almacenamiento y sistemas de comunicaciones (equipos de red). Dado a que estos componentes almacenan y procesan información (datos) críticos para sus clientes, está equipada con altos sistemas de seguridad, arie acondicionado para su correcta refrigeración, sistemas de incendios especiales, entre otros.

Los componentes de un centro de datos podrían clasificarse en tres:

- Infraestructura de red: La cual conecta los servidores, el almacenamiento y la conectividad externa a donde se encuentren los clientes.
- Infraestructura de almacenamiento: La cual guarda los datos.
- Recursos Informáticos: Estos son el motor de los centros de datos. Se encargan de aportar el procesamiento, la memoria, el almacenamiento local y la conectividad de red.

Entrando más en detalle, se podría decir que los elementos comunes en los centros de datos son:

1. **Hardware de computación**

   Unidades de informáticas, almacenamiento de datos y otro tipo de hardware.

2. **Racks**

   El hardware en un centro de datos normalmente esta montado en Racks (estanterías metálicas) para así poder aprovechar al máximo el espacio del lugar físico. Los Racks normalmente quedan pegados al techo, dejando únicamente espacio para meter el cableado y el sistema de refrigeración.

3. Infraestructura tecnológica

Dispositivos de red y seguridad que brindan servicios básicos y necesarios para los centros de datos tales como la conectividad a Internet.

4. Conectividad

Los centros de datos normalmente tienen multiples conexiones de fibra óptica a Internet proporcionadas por distintas empresas proveedoras, para así cercionarse de que siempre dispondrán de conexión.

5. Instalaciones

Los edificios donde se encuentran los centros de datos están expresamente diseñados para eso. Por ejemplo, la altura de los techos será acorde con los requisitos para los Racks y los sistemas instalados en él mismo, como sistema de refrigeración, cableado, etc.

6. Lugar

Un centro de datos requiere un sitio con conexiones a la red, tanto eléctrica como de Internet e infraestructura física como carreteras. Si el centro de datos se sitúa en un lugar donde el clima es frío, podría reducir los costes de refrigeración.

La proximidad a las zonas comerciales, clientes, empleados y servicios también juegan un papel importante en la selección de un sitio apropiado.

7. Energía

Cada máquina en el centro de datos debe de tener dos conexiones a la corriente, siempre y cuando el centro de datos tenga múltiples conexiones a la red. Debido a esto, la infraestructura eléctrica puede llegar a ser bastante compleja con características muy determinadas de distrubucion, derivación de energía y conmutación.

8. Sistema de cableado

Un sistema para administrar la gran longitud de cables que conectan cada máquina en un centro de datos a energía, redes, dispositivos y recursos. Es común que en los centros de datos el suelo se encuentre elevado para asi facilitar el acceso al cableado. Alternativamente, algunos sistemas de cableado cuelgan del techo.

9. UPS

Del inglés “Uninterruptible Power Source Systems”, sistema de fuente de energía ininterrumpible, el cual brinda protección en caso de cortes de corriente inesperados, sobretensiones o caídas de tensión.

10. Sistemas de producción de energía

Un sistema de producción de enrgía a modo de respaldo, como un generador con combustible. También es común que los centros de datos tengan un sistema de paneles solares en el techo o cerca. Por ejemplo, unos de los centros de datos de Google, tiene un parque eólico muy próximo, al cuál le compra la energía.

11. **Sistemas de refrigeración**

    Una parte esencial y tremendamente importante cuando hablamos de los centros de datos, ya que un sobrecalentamiento del hardware podría accarear en una rotura permanente del elemento y por lo cual una perdida de datos. Estos sistemas enfrían el hardware y proporcionan calefacción, ventilación, aire acondicionado, humidificación y deshumidificación para la instalación. La refrigeración es un elemento importante para la eficiencia de un centro de datos. Los centros de datos suelen estar diseñados específicamente para monitorizar la potencia requerida para el enfriamiento de los equipos.

12. **Protección antincendios.**

    Sistemas para la protección contra incendios, como detectores de humo y un sistema de rociadores contra incendios. También se suelen utilizar sistemas como barreras ignífugas para proteger los equipos.

13. **Meet-me Room**

    Area designada a las compañías de telecomunicaciones, donde físicamente van a conectar sus redes e intercambiar datos.

14. **Seguridad Física**

    Los centros de datos están diseñados teniendo muy en cuenta la seguridad, por ello tienen muy pocas ventanas y con sistemas como los Man-Trap para proteher el control de acceso.

    Por lo general están monitorizados con cámaras y pueden tener guardias de seguridad.

15. **Centro de Control de la Red**

    En inglés denominado NOC (Network Operations Center). Es una sala para el personal de operaciones con herramientas para monitorear, administrar, mantener y proteger los recursos informáticos. El Centro de Control de la red es el primer responsable de la gestión y control de incidencias y problemas dentro del centro de datos.

1.2 Como trabajan las empresas CPD

Los datos almacenados en el Centro de Datos pueden pertenecer a una sola empresa que tenga su propio centro de datos. Sin embargo, lo más común es que el Centro de Datos pertenezca a una empresa de servicios de informática y telecomunicaciones**,** que se encarga de custodiar la información de sus clientes. En este último caso, la empresa que gestiona el Centro de Datos dispone de equipos y espacio de almacenamiento suficiente para tratar millones de datos.

Las empresas responsables de estos Centros de Datos se encargan tanto del mantenimiento de los servidores como de garantizar una calidad de servicio a gusto del usuario como de la seguridad de sus datos almacenados en dichos servidores.

Estas empresas toman ventaja y hacen negocio a través de las pequeñas y medianas empresas, ya que estas son las que suelen contratar sus servicios.

A las pequeñas y medianas empresas economincamente hablando les merece la pena contratar este servicio, ya

que el montaje de la instalación y el propio manteminiento supone una gran inversión.

Al contratar los servicios de un Centro de Datos, las empresas lo que hacen es "alquilar" físicamente un numero de servidores dentro de un Clúster o incluso varios Clústers, dependiendo de la información que necesiten almacenar y procesar, de esta manera si hay algun fallo de hardware en un servidor concreto, la información no se perdería ya que se encontraría dentro de otro de ellos. Dependiendo de la calidad de servicio contratada, la velocidad de respuesta será mayor o menor.

1.3 Problemas que puede tener un Centro de Procesamiento de Datos

1.3.1 Problemas generales

Los CPD trabajan para poder dar un servicio rápido, fiable y que no se vea interrumpido por muchos factores externos que se encuentran presentes. Puede haber fallos físicos como la ruptura de hardware o fallos de comunicación, cortes de corriente, congestiones en la red y otros muchos problemas que los CPD deben resolver al instante para asi no perder su calidad de servicio. Debido al servicio que estos ofertan, estos tienen que funcionar las 24h del día, lo cual puede dar lugar a varios posibles problemas.

También hay que tener en cuenta la necesidad de los CPD de tener mas capacidad según crece la demanda. La necesidad de poder escalar las redes que posee para poder aumentar el ancho de banda y así mantener la fiabilidad es muy importante en estos centros. Lo mismo ocurre con los servidores, los cuales pueden ser escalados para así poder aumentar su funcionalidad. En estos casos las redes existentes necesitan ser capaces de manejar la congestión para asi poder controlar el flujo de los datos correctamente, y esta red solo será igual de rápida que sus componentes mas lentos. Además los CPD tienen que cumplir con la, SLA (Service Level Agreement), en castellano, Calidad de Servicio, lo que accarea los temas de rendimiento y de tiempo de respuesta, otros dos problemas con los que los CPD han de lidiar.

Pueden ocurrir numerosos numerosos fallos, lo cual hay que tener en cuenta. Los servidores y el equipo se pueden estropear. El cableado puede dejar de funcionar correctamente, la alimentación eléctrica se puede cortar por muchas razones. Por todas estas posibles ocurrencias es indispensable que todo este correctamente monitorizado, para así, nada más que el fallo ocurra sea notificado a los miembros del equipo que mantiene todo el sistema, y de esta manera pueda ser gestionado y solucionado rápida y eficazmente.

Un plan de recuperación de desastres es también importante en el caso de un fallo grave, pero los fallos leves también han de ser controlados.

Por estas razones, todo el sistema está configutado para desviar el tráfico de datos en caso de que los servidores o el equipamiento de red fallen en una zona. El tráfico también puede ser dividido para así distribuirlo y evitar congestiones y grandes colas (cuellos de botella). También se realizan y mantienen copias de seguridad y sistemas repetidos para que en caso de que haya una caída en la red todo pueda seguir funcionando hasta que se solucione el problema.

Normalmente los CPD también tienen dos proveedores de servicios de Internet, para así dar redundancia a la salida y a la entrada de datos. De esta manera el flujo de datos puede ser enviado a uno u otro proveedor según se necesite. El equipamiento hardware y el software han de actualizarse periódicamente para que no quede obsoleto. Y los componentes y el software más antiguos también tienen que ser soportados por el sistema hasta que puedan ser reemplazados. Todo el CPD debe estar diseñado de manera que todos los cambios y actualizaciones que se realicen, se hagan de manera rápida y sencilla.

Como también se ha mencionado anteriormente, estos centros suelen tratar con datos privados que han de ser protegidos de posibles ataques externos. Por esta razón se pueden necesitar puertas con cerraduras y sistemas de acceso electrónico, alarmas, personal de seguridad, etc. Algunas veces ni siquiera las compañías revelan

abiertamente quienes son sus CPD proveedores. Tmbien es necesario equipamiento de seguridad de red como pueden ser Firewalls, Proxies, etc.

1.3.2 Problemas de eficiencia energética

En los tiempos que corren, está habiendo un Bom en la construcción de centros de datos, cada vez más grandes y mas sofisticados, los cuales demandan las mejores tecnologías para así poder garantizar su funcionamiento y mejorar su eficiencia. Por ello, el consumo energético en estos centros también incrementa exponencialmente.

Ya sea impulsado por ahorrar costos, reducir la huella de carbono o ambos, los esfuerzos para reducir el consumo de energía en los CPD se ven dificultados por la adopción continua de tecnologías que consumen más energía, por lo cual generan más calor.

Los CPD consumen entre el 8 y el 35 por ciento de la energía total utilizada por empresas no manufactureras, lo que hace que los CPD sean el contribuyente más grande y mas concentrado a la huella de carbono empresarial, según el Uptime Institute.

Un CPD no es solo hardware y telecomunicaciones, como ya se ha citado anteriormente, cuenta con grandes instalaciones eléctricas, de refrigeración, contra incendios e iluminación entre otras muchas. La energía consumida en los CPD, no solo se centra en ellas mismas, es decir, en todo lo mencionado anteriormente, que ya es bastante, sino que gran parte del consumo energético en estos centros corresponde a los sistemas de refrigeración, parte que se convierte en objetivo central para la reducción de sus costes energéticos, y por lo tanto de su eficiencia.

La refrigeración de los CPD va completamente de la mano con el consumo energético y por tanto si se llega a tener una correcta refrigeración, se obtendrán unos ahorros energéticos importantes.

Uno de los problemas principales y que más preocupa a sus administradores es la producción de calor de los equipos que componen un centro de datos. El exceso de calor de estos servidores afecta negativamente al rendimiento del equipo y acorta su vida útil, además de suponer un peligro en el caso de alcanzar niveles muy elevados. Por este motivo es esencial el diseño de un correcto sistema de refrigeración en los CPD.

En el diseño del sistema de refrigeración de un CPD es fundamental el dimensionamiento del mismo, por lo cual es necesario entender bien la cantidad de calor producida por los equipos informáticos, junto con el calor que producen otras fuentes de calor que habitualmente están presentes como el cableado, la iluminación, las personas, la temperatura exterior,etc.

Normalmente en los CPD, la distribución de la carga térmica suele rondar:

- 70% la carga de los equipos informáticos.
- 9% la iluminación.
- 6% a la distribución de la alimentación.
- 2% a las personas.

A parte de la eliminación de calor, esos sitemas de aire acondicionado también están diseñados para regular la humedad en el ambiente. En la gran mayoría de sistemas de refrigeración, la función de enfriamiento del aire, causa por consiguiente una condensación de vapor de agua, por lo cual una perdida de humedad en el ambiente. Por esto es necesaria una humidificación suplementaria para mantener el nivel deseado. Esta humidificación suplementaria crea una carga de calor adicional en la unidad de refrigeración, disminuyendo

notoriamente la capacidad de enfriamiento de la unidad y requiriendo de un sobredimensionamiento.

También es importante el diseño de la red de conductos del aire o el falso suelo ya que afecta a la uniformidad de la temperatura dentro del CPD y en consecuencia al rendimiento global del sistema. La correcta implantación de un sistema de distribución de aire modular, unido a una acertada estimación de la carga térmica, puede reducir considerablemente la configuración del diseño en sí del CPD.

En los grandes CPD que normalmente oscilan entre los 500 m² y 1000 m², la densidad promedia del local es de 1500 W/m² donde los sectores de comunicación están aproximadamente en 300 W/m² y las zonas de alta densidad pueden llegar a los 4000 W/m². A esto habría que añadir también la refrigeración de las salas eléctricas, oficinas, talleres, etc. Por norma general se llega a capacidades de 3 a 5 MW de frío en un CPD, lo cual se puede ver claramente reflejado en el consumo eléctrico. Normalmente los sistemas suelen ser de agua helada, no sólo por el consumo eléctrico sino también por la flexibilidad que proporciona en su aplicación por el bajo volumen de refrigerantes en uso.

Este será el tema fundamental en este proyecto ya que toda la simulación realizada se utilizara para mejorar la eficiencia en los CPD.

2 HERRAMIENTA A UTILIZAR: MATLAB®

Millones de ingenieros y científicos de todo el mundo usan MATLAB® para analizar y diseñar los sistemas y productos que transforman nuestro mundo. MATLAB está presente en sistemas de seguridad activa de automóviles, naves espaciales interplanetarias, dispositivos de monitorización de la salud, redes eléctricas inteligentes y redes móviles LTE. Se utiliza para aprendizaje automático, procesamiento de señales, procesamiento de imágenes, visión artificial, comunicaciones, finanzas computacionales, diseño de control, robótica y muchos otros campos.

Matemáticas. Gráficas. Programación.

La plataforma de MATLAB está optimizada para resolver problemas científicos y de ingeniería. El lenguaje de MATLAB, basado en matrices, es la forma más natural del mundo para expresar las matemáticas computacionales. Las gráficas integradas facilitan la visualización de los datos y la obtención de información a partir de ellos. Una vasta biblioteca de herramientas (Toolboxes) integradas le permite empezar a trabajar inmediatamente con algoritmos esenciales para su dominio. El entorno de escritorio invita a experimentar, explorar y descubrir. Todas estas herramientas y funciones de MATLAB están probadas rigurosamente y diseñadas para trabajar juntas.

Expanda. Integre. Implemente.

MATLAB le ayuda a llevar sus ideas más allá del escritorio. Puede ejecutar sus análisis en conjuntos de datos de mayor tamaño y expandirse a clústeres y nubes. El código de MATLAB se puede integrar con otros lenguajes, lo que le permite implementar algoritmos y aplicaciones en sistemas web, empresariales o de producción.

Características principales

- ***Lenguaje de alto nivel para cálculos científicos y de ingeniería***
- ***Entorno de escritorio optimizado para la exploración iterativa, el diseño y la solución de problemas***
- ***Gráficas para visualizar datos y herramientas para crear diagramas personalizados***
- ***Aplicaciones para ajustar curvas, clasificar datos, analizar señales, ajustar sistemas de control y muchas otras tareas***
- ***Toolboxes complementarias para una amplia variedad de aplicaciones científicas y de ingeniería***
- ***Herramientas para crear aplicaciones con interfaces de usuario personalizadas***
- ***Interfaces para C/C++, Java®, .NET, Python, SQL, Hadoop y Microsoft® Excel®***
- ***Opciones de implementación libres de derechos para compartir programas de MATLAB con los usuarios finales***

[3]

3 MODELO DINÁMICO DE CENTRO DE PROCESAMIENTO DE DATOS

Los CPD están compuestos por unidades aisladas térmicamente en las que se encuentran los Racks de servidores de forma perimetral. Estas unidades aisladas pueden ser aisladas en frio (Cold Aisle) o en calor (Hot Aisle).

En las unidades tipo Cold Aisle, entra el aire frio del CRAC por el suelo, circula por un pasillo aislado donde los Racks toman el aire gracias a unos ventiladores que hacen pasar el aire frío por los servidores, para posteriormente salir al exterior donde se unen con el aire de otras unidades que circulan de vuelta al CRAC

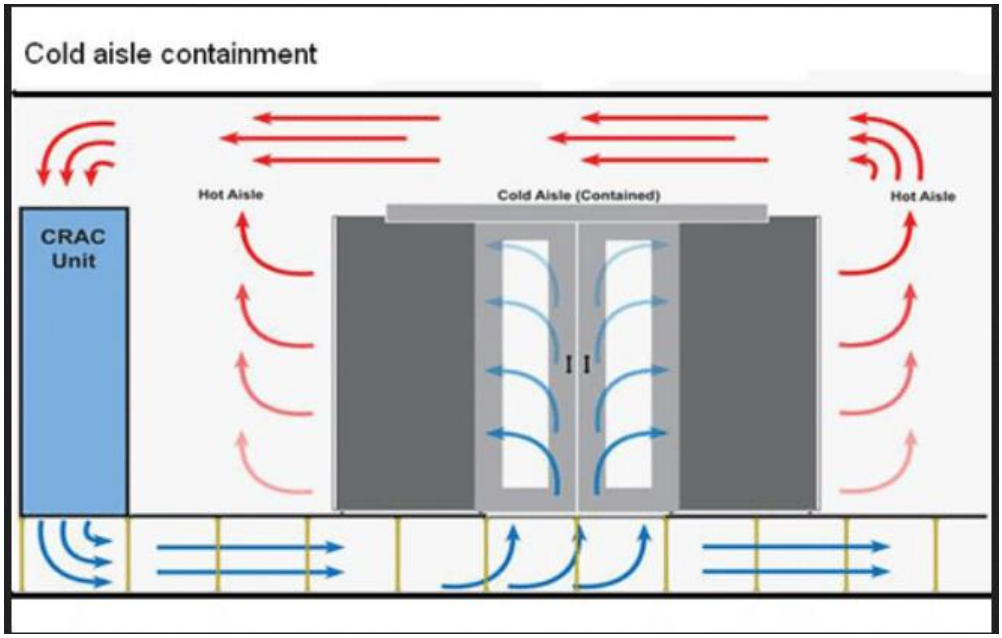

Figura 1. Esquema del Cold Aisle [1]

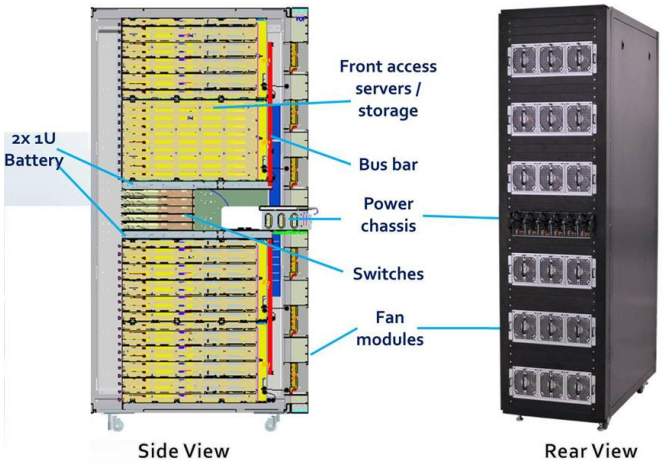

Figura 2. Sistema de ventiladores en el servidor [2]

2.1. Gestión de tareas

La gestión de las tareas en un servidor se hace mediante un sistema de gestión que manipula el nivel de tensón y la escala de frecuencia (DVFS) del servidor con el fin de mejorar su eficiencia energética, dependiendo de la carga de trabajo de los servicios, que se gestionan a través de máquinas virtuales (VM).

Como ya se ha explicado anteriormente, el CPD da servicio a unos usuarios (Cloud Users), que alquilan un cierto número de servidores virtuales, lo que sería la capacidad de cáculo, a un proveedor en la nube (Cloud Provider). El sistema de gestión será el que le asignará o gestionará los servidores reales.

El proveedor de servicios se compromete a aportar una calidad de servicio al usuario (QoS), en forma de un tiempo máximo de respuesta y una fiabilidad del servicio en forma de restricciones blandas en temperaturas de servicio.

2.2. Modelado del consumo de un servidor

El CPD está formado por un número de servidores M que los J ususarios pueden alquilar. Cada usuario j tiene reservado un total de servidores M_j para que lleven a cabo las peticiones de servicio que el usuario necesite (tareas). $L_j(t)$ será el número de peticiones que debe gestionar el CPD por unidad de tiempo para para un usuario j.

Los servidores van a tener tres estados, los cuales irán cambiando para así optimizar la realización de tareas o peticiones evitando un consumo innecesario, estos estados serán: *ocupado*, *ocioso* o *apagado*. Cuando el servidor se encuentra en estado *apagado,* el servidor estará inhabilitado, es decir, no podrá realizar ninguna acción. Si el estado del servidor es *ocioso,* quiere decir que el servidor se encuentra encendido, con lo cuál consumiendo energía, pero sin realizar ninguna acción, en este caso, el servidor suele estar esperando una orden para trabajar o en caso de no tener ninguna tarea que procesar apagarse. Cuando el servidor se encuentra en estado *ocupado,* quiere decir que en ese momento esta procesando alguna petición o tarea.

Cada usuario tendrá un número de servidores operativos, m_j. Por lo cual, el número de servidores apagados

será $M_j - m_j$. Entonces, el número de servidores operativos tiene un límitite, y este es el numero de servidores contratados por el cliente

$$0 \leq m_j \leq M_j$$

Esta es una variable manipulable, y se utilizará con el fin de mejorar la eficiencia energética del sistema.

Para una cantidad determinada de peticiones L_j, y un número de servidores operativos m_j, el sistema de gestión de los servidores, reparte las tareas por los diferentes servidores.

Una vez se asigna la tarea a un servidor, este la relaiza, lo cual le toma un tiempo en función de la complejidad de la tarea y de la velocidad de procesado del propio servidor. Esta velocidad se puede variar manipulando la frecuencia de operación del microprocesador, de forma que cuanto mayor sea la frecuencia (dentro de los límites operativos del servidor), mayor será la velocidad del procesado y menor será el tiempo que tarda en completarla.

Este aumento en la frecuencia también acarrea un aumento de potencia y por consiguiente un aumento del calor dispipado por la electrónica, lo que implica en un aumento de la temperatura del servidor que esté operando.

Lj/mj indica el número de peticiones a realizar por unidad de tiempo. Mientras mas grande sea esta tasa, mayor será la carga de trabaho por servidor y mayor será el tiempo de procesado. Para mejorar este tiempo de procesado, y por ende mejorar la calidad de servicio, se podrían aumentar el número de serviodres operativos dentro del margen admisible (M_j), o aumentar la frecuencia de funcionamiento, o ambas a la vez, eso sí, en ambos casos conllevaría a un aumento del consumo energético del servidor.

2.3. Modelado dinámico de los servidores

Se asume que para un usuario j el sistema de gestión de tareas mantiene una única cola en la que las tareas pendientes a realizar se iran almacenando una tras otras, a la espera de que se le asigne un servidor en estado *ocioso* disponible entre los m_j operativos.

También se asume que las peticiones vienen descritas por un tiempo entre llegadas gestionado por una distribución exponencial, cuya función de densidad viene dada por

$$f(t) = \frac{1}{T_a} e^{-\frac{t}{T_a}}$$

Siendo T_a el tiempo medio.

Siendo así, se sabe que el número de eventos que llega por unidad de tiempo T, L_j, sigue una distribución de probabilidad de Poisson. De esta manera, la distribución de probabilidad de que lleguen n peticiones en T segundos, viene dada por

$$f(n) = \frac{(\frac{T}{T_a})^n \, e^{-\frac{T}{T_a}}}{n!}$$

Cuyo valor medio es T/T_a. Llamando Lmj a la media del numero de tareas o peticiones por unidad de tiempo y siendo T la unidad de tiempo, se obtiene que $T/T_a = Lmj$ por consiguiente $T_a = T/L_{mj}$.

Por otra parte, el tiempo de servicio del servidor i se asume que viene descrito por una función de densidad exponencial, la cual viene dada por

$$f(t) = \frac{1}{T_{eij}}\, e^{-\frac{t}{T_{eij}}}$$

Cuyo tiempo medio de ejecución es T_{eij}.

La velocidad de procesado de cada servidor i viene dada por su frecuencia s_{ij} (medido en términos de instrucciones por segundo). Así, el tiempo medio de servicio será $T_{eij} = K/s_{ij}$, siendo K una media de la complejidad (en numero de instrucciones) del servicio que solicita el usuario.

La gestión de los servicios del ususario se modelará a través de una cola M/M/m_j, en la que el número de recursos es el numero de servidores operativos m_j. De este modo, el tiempo entre llegadas de peticiones de servicio o tareas sigue una distribución exponencial de media T/L_{mj}, siendo T la unidad de tiempo, y el servidor i-ésimo tiene un tiempo de servicio que sigue una distribución exponencial de media $T_{eij} = K/s_{ij}$.

2.4. Calidad de servicio

La calidad de servicio en un CPD se mide como el tiempo que tarda en responder el sistema ante la demanda de una petición de servicio, es decir, el tiempo que tarda en completarse una petición o tarea desde que esta llega al sistema de gestión hasta que esta se da por finalizada. Lo cual sería la suma del tiempo de espera en la cola mas el tiempo de procesado de dicha petición.

Para poder cumplir con un buen funcionamiento del CPD, se imponen restricciones en la calidad del servicio, medido en el tiempo medio de servicio T_{sj}.

$$T_{sj} = \frac{1}{\sum_{i=1}^{m_j} \frac{1}{T_{eij}} - L_{mj}}$$

Por lo cual, siendo D_j el tiempo máximo de servicio admisible, la restricción impuesta sería

$$T_{sj} \leq D_j$$

2.5. Modelado de la potencia de los servidores

Mientras el sistema esté trabajando, cada servidor i asociado al usuario j tendrá un consumo energético y una potencia disipada en forma de calor. Esta potencia disipada por el servidor dependerá del estado en el que este se encuentre, es decir, *apagado*, *ocioso* u *ocupado*. Un servidor *apagado* no disipa ninguna potencia, al arrancarse consume una potencia, pero tarda un tiempo en estar operativo para realizar tareas, este tiempo será T_{arr} (Como todos los servidores son iguales, se asume que este tiempo de arranque es el mismo para todos ellos). Cuando el servidor se encuentra operativo, cuando esta en estado *ocioso*, este consume una potencia minima asociada a la electrónica básica, sistemas de almacenamiento y alimentación. Cuando el servidor se encuentra en estado *ocupado*, además de esta potencia, disipa la potencia asociada al uso de los recursos hardware necesarios para ejecutar la tarea, esta potencia será mayor cuanto mayor sea la frecuencia de la operación del servidor s_{ij}.

De este modo, el consumo de un servidor i asociado al usuario j viene dado por

$$p_{ij} = \begin{cases} a_2 & si\ está\ ocioso\ o\ arrancando \\ a_1 + a_2 & si\ está\ ocupado \end{cases}$$

El consumo marginal del servidor será a_1, sin carga de trabajo, y este a su vez dependerá de la frecuencia del servidor, de manera que $a_1 = C_f \cdot s_{ij}$, siendo C_f una constante. El consumo del servidor sin carga de trabajo

será a_2, siendo muy semejante al consumo durante el tiempo de arranque, por lo que se considerarán iguales.

De está manera el consumo de los servidores vendrá dado por

$$p_j(t) = a_1 \cdot m_{oj}(t) + a_2 \cdot m_j(t)$$

Donde $m_{oj}(t)$ será el número de servidores ocupados en el instante *t* y $m_j(t)$ el número de servidores operativos, en el mismo instante, para el usuario *j*.

2.6. Modelado térmico de los servidores

Para modelar la temperatura de cada servidor, se va a realizar un balance térmico de cada uno de ellos, asumiendo que la temperatura de la unidad aislada en frío es θ_c. La temperatura del servidor *i* del usuario *j* será θ_{ij} y siendo K_t la capacidad térmica del servidor, C_p la capacidad térmica del aire y $q_a$el caudal de aire, su dinámica vendrá dada por

$$K_t \frac{d\theta_{ij}}{dt} = C_p \cdot q_a\left(\theta_c - \theta_{ij}\right) + p_{ij}$$

Los servidores tienen un límite de temperatura media a un cierto valor, aunque este valor puede ser superado durante un corto intervalo de tiempo en un momento determinado, en forma de restricción blanda. Las temperaturas máximas de operación también pueden considerarse como restricciones duras.

2.7. Modelado de la máquina de refrigeración

La variable manipulable dentro del sistema de control de una maquina de refrigeración será la temperatura de consigna del aire de salida de la misma, que en este caso será θ_r .

La máquina de refrigeración será modelada como un sistema de primer orden de ganancia estática unidad

$$\tau \frac{d\theta_c}{dt} = \theta_r - \theta_c$$

La potencia consumida por la máquina de refrigeración es

$$P_{CRAC}(t) = \frac{P(t)}{CoP(\theta_c(t))}$$

Siendo *P* la potencia consumida por el CPD y CoP(·) la función de desempeño de la máquina, la cual es una función parabólica

$$CoP(\theta) = b\theta^2 + c\theta + d$$

Donde los coeficientes b, c y d dependerán de la máquina de refrigeración y los provee el fabricante.

2.8. Índice de desempeño

Se suele tomar como índice de desempeño el PUE (Performance Usage Efficiency) que viene dado por la siguiente expresión

$$PUE(t) = \frac{E_C(t)}{E_C(t) + E_{CRAC}(t)}$$

Donde E_C será la energía consumida por los servidores y E_{CRAC} la energía consumida por el sistema de refrigeración

Siendo $E_C(t) = \int_0^t P(\tau)d\tau$ y $E_{CRAC}(t) = \int_0^t \frac{P(\tau)}{CoP(\theta_c(\tau))}d\tau$

2.9. Dimensionamiento de parametros

Todos los datos que van a emplearse en este modelo están extraidos del artículo de Fu, donde se muestran datos experimentales de un CPD, de donde se deducen los posibles parámetros físicos de nuestro modelo.

En este modelo se consideran un total de 1700 servidores, y se realizaran 4 aplicaciones que oscilan entre 1000 y 150000 tareas por segundo en cada aplicación.

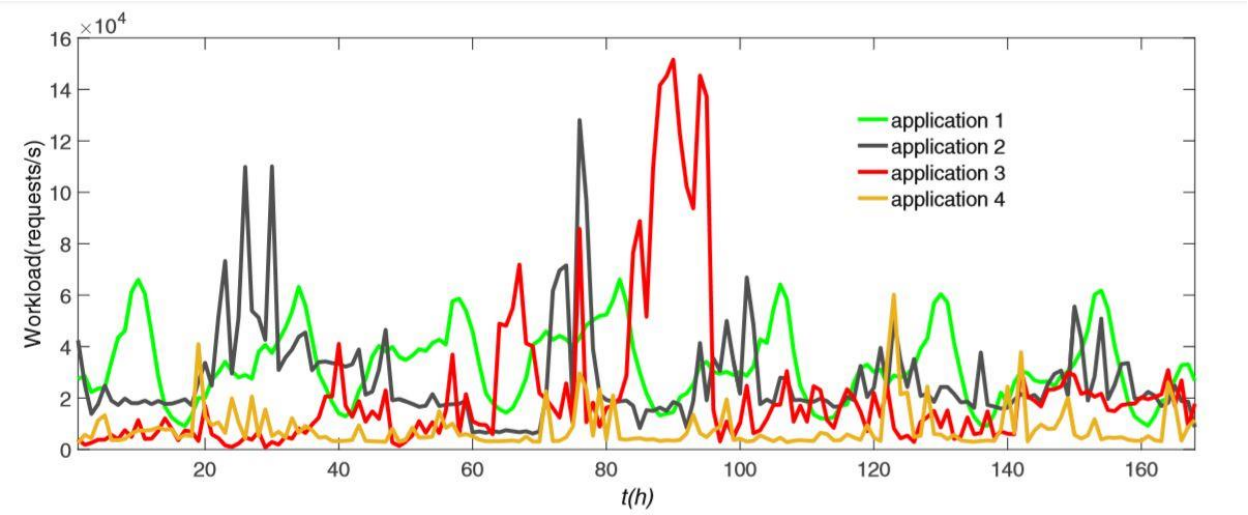

Figura 3. Peticiones en las 4 aplicaciones [1]

El total sería la suma de estas cuatro curvas. La media de la suma de las curvas se estima en 100000 (40000, 30000, 20000, 10000).

Se asume que el tiempo de procesado es de 0.01 segundos por tarea, que el consumo energético es:

$a_1 = a_2 = 40W$, la calidad del servicio tendrá un tiempo máximo $D_j = 0.05s$ y la temperatura de refrigeración será $15 \leq \theta_c \leq 25ºC$.

Los parámetros térmicos serían $C_p \cdot q_a = 1.6$ y $K_t = 1.6$ que será una constante de tiempo del orden de 1s y temperatura máxima de servicio $\theta_c + 50ºC$, a potencia máxima de 80W

Escalado a 10 servidores, la media de llegadas por segundo para nuestro modelo seria 580, la cual redondearemos a 600.

El tiempo de servicio debe ser menor que 0.0172 segundos. De este modo, la capacidad media de los 10 servidores ($\frac{10}{T_{serv}} > \frac{10}{0.0172} = 580$) es mayor que el número de peticiones por segundo. Finalmente tendríamos

que $T_{serv} = 0.01$ y $T_{lleg} = 0.0016$. De todos modos, para facilitar el desarrollo de la implementación del modelo en Matlab hemos multiplicado ambos tiempos por diez, resultando $T_{serv} = 0.1$ y $T_{lleg} = 0.016$ (segundos).

2.10. Definición de variables del modelo

Primeramente enunciaremos las variables manipulables de nuestro sistema, estas variables son aquellas que podemos cambiar para influir en los resultados y así poder obtener los resultados deseados.

- Lj (número de peticiones) esta variable originariamente es una perturbación, pero en el modelo implementado se utilizará como variable manipulable ya que es la que nos dara más o menos tiempo de simulación.
- mj que será el número de servidores operativos
- s_{ij} la frecuencia sería otra variable manipulable del sistema, pero no se tiene en cuenta en el modelo implementado para facilitar los cálculos.
- θ_r La temperatura de consigna del aire. En el modelo implementado será Tr

Variables fijadas por el sistema

- Mj número contratado de servidores
- θ_c La temperatura de la unidad aislada en frío, aunque es manipulable a través de θ_r. En nuestro modelo Tc

Dentro del modelo se encuentran muchas variables internas para el correcto funcionamiento del sistema que se irán explicando a lo largo del desarrollo de la implementación.

4 IMPLEMENTACIÓN DEL MODELO EN MATLAB®

Una vez explicada la funcionalidad de la herramienta a utilizar y definido el modelo dinámico del sistema, podemos pasar a la implementación del modelo

3.1. Implementación

Tendremos que tener en cuenta que el sistema esta modelado en milisegundos, ya que únicamente se utilizan números enteros, para así simplificar. También hay que tener en cuenta que en este modelo no se encuentran implementados ni el tiempo de arranque T_{arr}, ni la frecuencia s_{ij}, para facilitar la implementación del modelo en la herramienta.

Este modelo se va a implementar con un usuario y un máximo de 10 servidores (M_j = 10, número de servidores reservados)

La implementación comienza con la declaración de todas las variables que se utilizaran en el sistema.

```
clear all; close all

% coment=0;
rng(0)
%ESTAMOS TRATANDO TODO EN ms PARA SIMPLIFICAR
%Entradas

Lj = 900;   %Numero de peticiones
Mj = 10;    %Numero de servidores maximo
mj = 0;  %Servidores encendidos
mjhist=[];

%Historico de numero de servidores Idle
Nidleshist=[];
Nidleslhist=[];

%Historico de servidores ON
ONhist=[];

%Historico de la potencia total
Pjhist=[];

%Historico de la potencia en cada ciclo
Pijhist=[];

%Historico de la longitud de la cola
LongitudColahist=[];
```

Figura 4. Declaración de Variables 1.

```
%Potencias

a1 = 40;
a2 = 40;

%Térmicos

cpqa = 1.6; %calor especifico por caudal
Kt = 1000;
tau = 10000;

Tc = 20; %Temperatura de la unidad aislada
Tr = 20; %Temperatura de consigna del aire
Trhist=[];
%Tiempos. Pendientes de ajuste

T = 17; %Tiempo medio entre llegadas de peticiones
P = 17*5; %Tiempo que tarda en procesar una petición


%Construimos los intervalos en los que van a aparecer las peticiones
%almacenados en un array
IPeticiones = ceil(exprnd(T*ones(1,Lj)));

%Tiempo que tarda en procesar cada petición almacenado en un array
Tprocesado = ceil(exprnd(P*ones(1,Lj)));
```

Figura 5. Declaración de Variables 2.

En nuestro modelo, las líneas de tiempo serán creadas a priori siguiendo las distribuciones exponenciales con los correspondientes tiempos de llegada y de procesado explicados anteriormente. Para esto, se ha creado un vector previo llamado *IPeticiones* que tiene un tamaño igual al número de peticiones *Lj* (Figura 5). Después, a través de un bucle *for* se creaun vector que va creciendo con las peticiones, acumulando los valores de llegada de las peticiones (Figura 6).

```
%Creamos la linea temporal de peticiones

Tacumulado = 0;

for aux = 1:Lj %recorremos el array 1 a Lj(peticiones)
    Tllegada (aux) = IPeticiones(aux)+Tacumulado;
    Tacumulado=Tllegada (aux);
end
```

Figura 6. Línea temporal.

A continuación, para simular los servidores, crearemos una matriz de tres filas y largo Mj (número máximo de servidores), donde la primera fila será el estado del servidor (apagado = -1, idle = 0, ocupado >1), la segunda será carga de la petición (tiempo de procesado asociado a la entidad) y la tercera la carga restante de la petición (tiempo de procesado que ira completándose prograsivamente) (Figura 7).

```
%Representación de los servidores, crearemos una matriz de tres filas y
%largo Mj, donde la primera será el estado del servidor (apagado =  -1,
%idle = 0,ocupado >1),la segunda será carga de la petición y la tercera la
%carga restante de la petición (Tprocesado)
servidores = zeros (3, Mj);

servidoresLibres = mj(1);   %En el instante inicial todos los servidores se
                            %encuentran libres
servidores(1,mj(1)+1:end) = -1; %Apaga todos los que no esten encendidos
                                %inicialmente, mj (servidores encendidos)
```

Figura 7. Creación de la matriz servidores.

Para gestionar las colas del sistema, se creará un vector que irá almacenando las entidades que vayan accediendo a la misma, también se creará un puntero de cola, el cual indicará su tamaño máximo y será necesario para el correcto funcionamiento de la misma. (Figura 8)

```
%Array para almacenar la cola
colaj = zeros(2,Lj);
pcolaj = Lj; %puntero de cola en Lj ya que es el ultimo elemento
```

Figura 8. Creación vector cola y puntero.

Seguidamente almacenaremos los tiempos de entrada y de salida al servidor por las peticiones, los cuales utilizaremos mas adelante para calcular el tiempo de servicio, también se crea una variable llamada Tserv, que se utilizará para calcular dicho tiempo.

```
%Almacenaremos los tiempos de entrada y de salida al servidor por las
%peticiones
Tin = zeros (1,Lj);
Tout = zeros (1,Lj);
Tserv = [];
```

Figura 9. Vectores Tin, Tout.

Creamos otro vector para almacenar ahí las temperaturas de los servidores, inicialmente será un vector de una fila llena de 20, de tamaño Mj, ya que esta será la temperatura inicial de los servidores. (Figura 10).

```
%Creamos un array para almacenar las temperaturas
Temp = 20*ones(1,Mj);
Temphist=[];
Tchist=[];
```

Figura 10. Vector de temperaturas de los servidores.

Después creamos otro vector, nos servirá para ir almacenando la potencia de cada servidor, este vector al inicio es un vector de ceros de longitud Mj. (Figura 11)

```
%Array para almacenar las potencias
Pij=zeros(1,Mj);
```

Figura 11. Vector de potencias.

Este vector también tiene un histórico para ir almacenando las potencias en cada iteración del bucle, declarado anteriormente, se puede observar en la Figura 4.

A continuación establecemos una condición inicial antes de iniciar el bucle *while*, que va a ser el bucle principal de nuestro modelo. Y abrimos el bucle *while* poniéndole como condición de salida que el tiempo de enfriado haya terminado, al finalizar el proceso, el sistema tiene un tiempo de enfriado, después de procesar todas las peticiones exigidas, al haber finalizado este enfriado, la variable *EnfriadoTerminado* pasa a valer 1, lo cuál hace que el bucle finalice. (Figura 12)

```
%Situación inicial
t = 1;
NumeroPeticion = 1;
PeticionesAtendidas = 0;
EnfriadoTerminado = 0;
Enfriando = 0;
tant=0;
inct=0;


while (EnfriadoTerminado ~= 1)
```

Figura 12. Inicio bucle while.

Dentro de este bucle *while*, empezaremos la a iterar planteando unas condiciones de encendido y apagado, de los servidores, predeterminadas, para plantear los escalones dependiendo de una variable llamada *t* que será nuestro tiempo, medido en milisegundos. (Figura 13)

```
if (t>=1000)&&(t<=1200)
    mj=1;
elseif (t>1200)&&(t<=1400)
    mj=2;
elseif (t>1400)&&(t<=1600)
    mj=3;
elseif (t>1600)&&(t<=1800)
    mj=4;
elseif (t>1800)&&(t<=2000)
    mj=5;
elseif (t>2000)&&(t<=2200)
    mj=6;
elseif (t>2200)&&(t<=2400)
    mj=7;
elseif (t>2400)&&(t<=3000)
    mj=8;
elseif (t>3000)&&(t<=5000)
    mj=9;
else
    mj=10;
end
```

Figura 13. Política de encendido de los servidores

Como se puede observar en este tramo de código, conformado por bucles *if else* desde 0 hasta 1000 milisegundos, va a haber 10 servidores operativos, ya que mj=10, después entre 1000 y 1200 bajan a únicamente 1 servidor, mj=1, después de esta brusca bajada los servidores vuelven a subir de uno en uno en los intervalos que se ven en la Figura 13, hasta llegar a 5001, donde vuelven a estar otra vez los 10 servidores operativos, mj=10.

A continuación estaría programado el cambio de Tr (escalón), que se realizaría en el instante t=5000 como se pruede apreciar en la Figura 14, este escalón esta creado también con un bucle *if else*, el que hace que cuando *t* sea mayor que 5000, *Tr* baje de 25ºC a 20ºC

```
if (t<5000)
    Tr = 25;
else
    Tr = 20;
```

Figura 14. Escalón en Tr.

Después de haber realizado el cambio de temperatura, se define la política de apagado de todos los servidores y la consecuentemente salida del bucle *while*, esto se realiza a través de tres bucles *if else,* en el primero se dice que si la variable *Enfriando* vale 1, es decir, si el enfriado ha finalizado, se apagan todos los servidores, en el segundo bucle, se dice que si todas las peticiones han sido atendidas (Lj) y el enfriado continua, que la variable Enfriando pase a valer 1, que se apaguen todos los servidores (mj=0) y que comience el tiempo de enfriamiento (TInicioE), en el tercer y último bucle, se dice que si el enfriado a finalizado y la resta del tiempo real, con la variable TInicioE es igual a 3000, es decir, si han pasado 3000ms desde el apagado de los servidores, hace que la variable EnfriadoTerminado pase a valer 1, la cuál es la condición de salida del bucle *while*(Figura12). (Figura 15).

```
if (Enfriando == 1) %si el enfriado a terminado,
        mj = 0;     %se apagan los servidores
end


if (PeticionesAtendidas == Lj)&& (Enfriando == 0)
    Enfriando = 1;
    mj = 0;
    TInicioE = t;
end

if (Enfriando == 1) && (t-TInicioE == 3000)
    EnfriadoTerminado = 1;
end
```

Figura 15. Apagado del sistema.

Seguidamente pasamos a empezar a meter elementos en la cola creada anteriormente, para ello creamos un bucle *if* en el diremos que si *t* (el tiempo actual) es igual al tiempo de llegada de la petición, la petición entraría en cola, con su correspondiente número de petición (NumeroPeticion), que sería como una especie de identificador, y con su tiempo de procesado (Tprocesado). Dentro de este bucle *if,* nos encontraríamos con otro, que iría incrementando el número de petición, hasta llegar a la última petición de la cola (Lj).

```
if (t == Tllegada(NumeroPeticion)) %si el tiempo es igual el tiempo de llegada de la peticion
    [colaj,pcolaj] = push (colaj, pcolaj,[NumeroPeticion;Tprocesado(NumeroPeticion)]); %Introduce en la cola la entidad
                                                                                       %con su correspondiente numero de entidad
                                                                                       %y tiempo de procesado(carga)
   if (NumeroPeticion <Lj) %Que siga avanzando hasta que llegue a la ultima petición
       NumeroPeticion = NumeroPeticion+1;
   end

end
```

Figura 16. Función if para meter elementos en la cola

Ahora determinaremos los servidores que se encuentran en los tres diferentes estados posibles, es decir, *apagados, ocupados* u *ociosos*. Para ello, lo primero que haremos será, ayudándonos de la función *find()*, encontrar los servidores que están *encendidos* (> -1), para ello, recorreremos la primera fila de la matriz *servidores* en busca de valores que sean > -1, con estos valores crearemos otro vector llamado servidoresON, el cual contiene los índices de los valores del vector > -1. Se realizará lo mismo para ver que servidores se encuentran en estado *apagado*, lo único que cambiaría es que en vez de buscar los valores > -1, buscaremos los valores = -1, y a este vector se le llamar servidoresOFF. (Figura 17)

Seguidamente, con la función size, la cual mide el tamaño del vector y te devuelve el número correspondiente a dicho tamaño, hayamos los tamaños de los vectores servidoresON y servidoresOFF y a estos dos escalares les llamaremos numeroON y numeroOFF respectivamente. (Figura 17)

Después utilizaremos el numero de servidores encendidos (numeroON), para ver que servidores hay que apagar o encender, para ello creamos la variable ON_OFF, esta será la diferencia de numeroON y mj (numero de servidores que quiero tener encendidos). Si la diferencia de estos dos escalares me devuelve un número negativo, quiere decir que necesitomos más servidores encendidos de los que hay, por lo cual enciendemos directamente los servidores, poniéndolos en estado *ocioso* a través del bucle *while* que se puede ver en la Figura 18.

Dando de resultado, la diferencia, un número positivo, lo que tendríamos que hacer es apagar los servidores que nos sobran, ya que el numeroON (servidores encendidos) sería mayor que mj (servidores que necesito encendidos). Para ello utilizaremos otro bucle *while*, pero antes habría que ver qué servidores podríamos apagar, ya que no pueden apagarse cuando están en medio de una tarea. Por esa razón, antes de entrar en el bucle *while* para apagarlos, crearemos una variable vectorial llamada *servidoresIDLE* con esta variable y la función *find*, y otra variable llamada numeroIDLE y la función *size*, haremos lo mismo que para encontrar los servidores apagados y encendidos, pero esta vez viendo qué números de la primera fila de la matriz *servidores* es igual a 0. (Figura 18)

```
%determinación de servidores apagados, ocupados o iddle
servidoresON = find(servidores(1,:) > -1);
servidoresOFF = find(servidores(1,:) == -1);

%contamos los encendidos
numeroON = size(servidoresON,2);
numeroOFF = size (servidoresOFF,2);
ON_OFF = numeroON-mj; %encendidos - necesitoencendidos
```

Figura 17. Determinación de estado de los servidores.

```
if (ON_OFF<0)
    aux=1;
    while (ON_OFF<0)
        servidores(1,servidoresOFF(aux)) = 0;
        aux = aux+1;
        ON_OFF = ON_OFF+1;
        servidoresLibres=servidoresLibres+1;
    end

elseif (ON_OFF>0)
    servidoresIDLE = find(servidores(1,:)==0);
    numeroIDLE = size(servidoresIDLE, 2);
    aux=1;
    while (ON_OFF>0) && (numeroIDLE>0)
        servidores(1,servidoresIDLE(aux))=-1;
        aux=aux+1;
        numeroIDLE=numeroIDLE-1;
        ON_OFF=ON_OFF-1;
        servidoresLibres=servidoresLibres -1;
    end
end
```

Figura 18. Apagado y encendido.

Ahora veremos donde se utiliza la función paralela pop, para sacar elementos de la cola, la condición de entrada al bucle *while* que lidera este tramo de código, es que haya servidores libres, y qy alguna petición en la cola, es decir, *pcolaj ~= Lj* entonces, se reduce en uno el número de servidores libres y se asigna un servidor a la petición, pero primero se comprueba si hay algún servidor libre, a través de otro bucle *while* y mientras haya elementos en la cola se iran itroduciendo en los servidores hasta que estos estén completos. Al meter la entidad o petición en el servidor, inicialmente, la carga restante será igual a la original. Finalmente anotaremos el tiempo de entrada de la petición al servidor. (Figura 19)

```
while (servidoresLibres>0 && pcolaj~=Lj)
    servidoresLibres=servidoresLibres-1;
    i=1;
    while (servidores(1,i)~=0)
        i=i+1;
    end
    [colaj,pcolaj,servidores(1:2,i)]=popCPD(colaj,pcolaj); %Al h
    servidores (3,i) = servidores (2,i); %El tiempo restante de
    Tin(servidores(1,i))= t;
end
```

Figura 19. Pop para sacar los elementos de la cola.

A continuación, haremos un recuento de los servidores que se encuentran en estado *ocioso*, creando la variable *Nidles,* esta, con la función *size* hará un recuento de dichos servidores, lo cuál usaremos mas adelante para el cálculo de la potencia. (Figura 20)

```
Nidles = size(find(servidores(1,:) == 0),2);
```

Figura 20. Recuento de servidores ociosos.

Posteriormente, recorreremos todas las columnas de la primera fila de la matriz servidor, utilizando un bucle *for* para ir reduciendo la carga de trabajo de cada petición, en una unidad por cada iteración del bucle, hasta que esta carga llegue al valor 0, entonces apuntaremos el tiempo de salida en la variable *Tout* e incrementaremos el número de servidores libres y de peticiones antedidas. Luego comprobaremos si queda algún servidor por apagar utilizando otro bucle *if* y la variable *ON_OFF*, creada anteriormente, en caso de que haya que apagar algún servidor, se apaga y se descontabiliza de los servidores libres, de esta manera evitaremos que se contabilicen como *ociososo* servidores que deberían estar *apagados* y también garantizaremos que no se apagará ningún servidor mientras se encuentre procesando. (Figura 21)

```
for i=1:Mj
    if (servidores(1,i)>0)
        servidores(3,i)=servidores(3,i)-1;
        if (servidores(3,i)==0)
            Tout(servidores(1,i)) = t;
            servidores(:,i)=[0;0;0];
            servidoresLibres=servidoresLibres+1;
            PeticionesAtendidas = PeticionesAtendidas+1;

               if (ON_OFF > 0)
               servidores(1,i)=-1;
               servidoresLibres=servidoresLibres-1;
               ON_OFF=ON_OFF-1;

               end
        end
    end
end
```

Figura 21. Reducción de la carga de la petición

Acto seguido comenzaremos ha hacer mediciones para obtener los resultados, como la longitud de la cola, la cual iremos guardando en cada iteración dentro de un histórico para poder representarla gráficamente.

El cálculo de la potencia total, en el cuál creamos una variable llamada *ON* la que se encargará de almacenar el número de servidores encendidos, utilizando la función *size(·)*, ya mencionada anteriormente. Para después restarle el número de servidores en estado *ocioso* ayudándonos de la variable *Nidles* (Figura20) para después multiplicar esos valores por su correspondiente potencia, que en este caso seria *a1* (a_1 en el modelo). Seguidamente utilizaremos el dato obtenido por la variable anteriormente mencionada *Nidles*, este valor será multiplicado por *a2* (a_2 en el modelo). La suma de ambos será igual a la potencia total (Pj). (Figura 21)

```
%Longitud de la cola
LongitudCola = Lj-pcolaj;

%Cálculo de la potencia total
ON = size(find(servidores(1,:)>=0),2);
Pj = a1*(ON-Nidles)+a2*Nidles;
```

Figura 22. Potencia total.

Acto seguido, continuaremos con la potencia por cada servidor, para el calculo de esta potencia utilizaremos tres funciones *find*, con esta función buscaremos en la primera fila de la matriz *servidores*, cuales son los servidores que se encuentran encendidos, a estos les daremos el valor *a1+a2* ($a_1 + a_2$ en el modelo), a los servidores que se encuentran en estado *ocioso* les daremos el valor *a2* (a_2 en el modelo) y a los que se encuentran *apagados*, les daremos el valor 0, estas potencias también se irán almacenando en un histórico para poder representarlas posteriormente. (Figura 22)

```
%Cálculo de la potencia por cada servidor
Pij(find(servidores(1,:)>=1))=a1+a2;
Pij(find(servidores(1,:)==0))=a2;
Pij(find(servidores(1,:)==-1))=0;
```

Figura 23. Potencia en cada servidor

A continuación pasaremos al cálculo de las temperaturas de cada servidor Temp (θ_{ij} en el modelo) y a la temperatura de la unidad aislada en frío Tc (θ_c en el modelo). Para ello comenzaremos haciendo un incremendo del tiempo, esto lo haremos guardando el tiempo en cada iteración como *tant* (tiempo anterior) y restandosela al tiempo real (t) en la siguiente iteración, así iremos obteniendo el incremento de tiempo de cada iteración para después utilizarlo en las fórmulas de las temperaturas, las cuales podemos observar en la Figura 23.

```
%incremento de tiempo
inct=t-tant;
tant=t;
 %Cálculo de las temperaturas
 Temp = Temp+inct*1/Kt*((cpqa)*(Tc-Temp)+Pij);
 Temphist=[Temphist;Temp];
 Tc=Tc+inct*1/tau*(Tr-Tc);
```

Figura 24. Cálculo de las temperaturas.

Después de las temperaturas lo terminaríamos con el almacenamiento de históricos, donde vamos creando esas filas apiladas con toda la información que vamos guardando, el calculo del tiempo de servicio ($Tserv$), que sería la diferencia del tiempo de salida con el tiempo de entrada de la petición del servidor ($Tout - Tin$) y el tiempo medio de servicio, que se realizaría con la función *mean(·)*, la cuál haya la media de los componentes de un vector, en este caso del vector $Tserv$.

```
    %Almacenamiento de historicos
    Tchist=[Tchist;Tc];
    Trhist=[Trhist;Tr];
    mjhist=[mjhist,mj];
    ONhist=[ONhist,ON];
    LongitudColahist=[LongitudColahist,LongitudCola];
    Pjhist=[Pjhist,Pj];
    Pijhist=[Pijhist;Pij];

   %Tiempo de servicio
   Tserv = Tout-Tin;

   %Tiempo medio de servicio
   Tmserv=mean(Tserv);
   t=t+1;
end
```

Figura 25. Historicos y tiempo de servicio.

5 RESTULTADOS Y ENSAYOS

5.1 Respuesta de las temperaturas a la alteración de servidores operativos

Vamos a realizar una simulación encendiendo y apagando los servidores para ver como reaccionan las temperaturas. Para ello vamos a simular con una llegada de 900 peticiones (Lj=900), y con una mj que variará con el tiempo, de 0 a 1000 ms habrá 10 servidores encendidos, de 1000 a 1200 ms bajaremos a mj=1, de 1200 a 1400 ms subiremos a mj=2, de 1400 a 1600 ms a mj=3, de 1600 a 1800 ms a mj=4, de 1800 a 2000 ms a mj=5, de 2000 a 2200 ms a mj=6, de 2200 a 2400 ms a mj=7, de 2400 a 3000 ms a mj=8, de 3000 a 5000 ms mj=9, y a partir de 5000ms será mj=10.

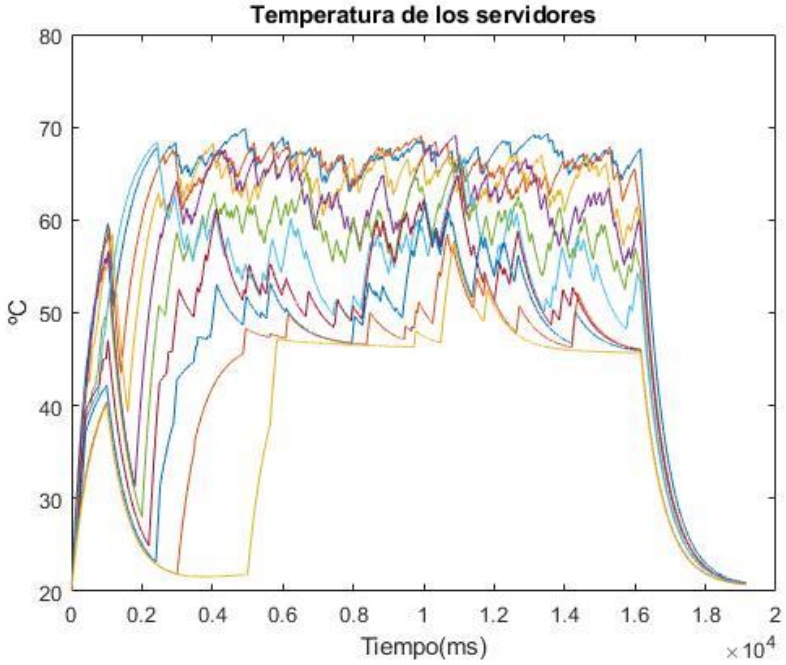

Figura 26. Gráfico de temperaturas de los servidores.

En este gráfico de las temperaturas de los servidodes con respecto al tiempo podemos ver perfectamente la relación del número de servidores activos con la temperatura, como se puede apreciar, al principio, al estar los 10 servidores en activo, las temperaturas empiezan a incrementar, y cuando llega al punto $0{,}12\ x\ 10^4$, podemos ver como todas las temperaturas de los servidores excepto de uno empiezan a bajar, ya que han dejado de estar encendidos y por tanto han dejado de consumir y por ende de disipar calor. Después vamos viendo como progresivamente van volviendo a subir las temperaturas de todos conforme los vamos encendiendo.

5.2 Respuesta de la longitud de cola a la alteración de servidores operativos

Vamos a realizar el mismo ensayo, con los mismos datos del apartado anterior, pero esta vez vamos a ver la relación que tiene el encendido y apagado de servidores con la longitud de cola.

Analizaremos la relación a través de un gráfico sacado de la simulación.

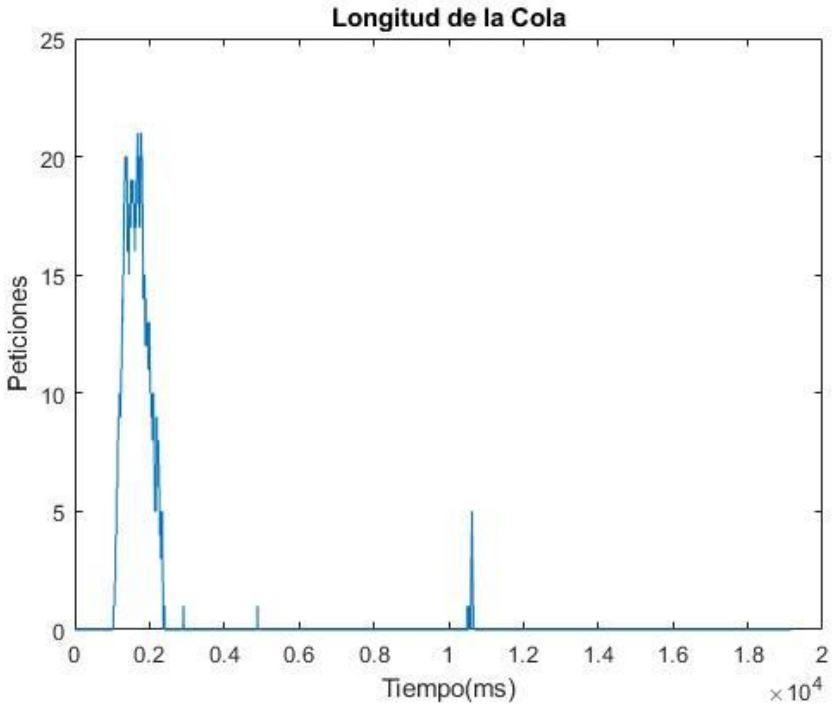

Figura 27. Gráfico de la longitud de la cola

En este gráfico podemos observar algo parecido pero al contrario que en la gráfica anterior, vemos como en el intervalo $[\,0,\ \ 1.2x10^4]$ no se forma ningún tipo de cola, sin embargo nada más que sobrepasa el punto $1.2x10^4$, la cola empieza a crecer rápidamente, y después vuelve a bajar rápidamente de nuevo conforme se van encendiendo los servidores que antes estaban apagados. Con esto podemos ver la relación directa que existe entre la temperatura y la cola por ejemplo, relacionadolo a través de la alteración de los servidores operativos.

3.2. Tr, Tc y su ganancia estática unidad en un sistema de primer orden

En base a este ensayo también podemos suponer que el tiempo en el que se encuentra el escalon de bajada de temperatura, ahora mismo situado en el instante t=5000 para disminuir Tr de 25ºC a 20ºC, debería de ser en un instante mucho mas lejano que en el se encuentra, ya que al ser un sistema de primer orden de ganancia estática unidad, Tc debería de alcanzar el escalón, y sin embargo se queda en los 22ºC

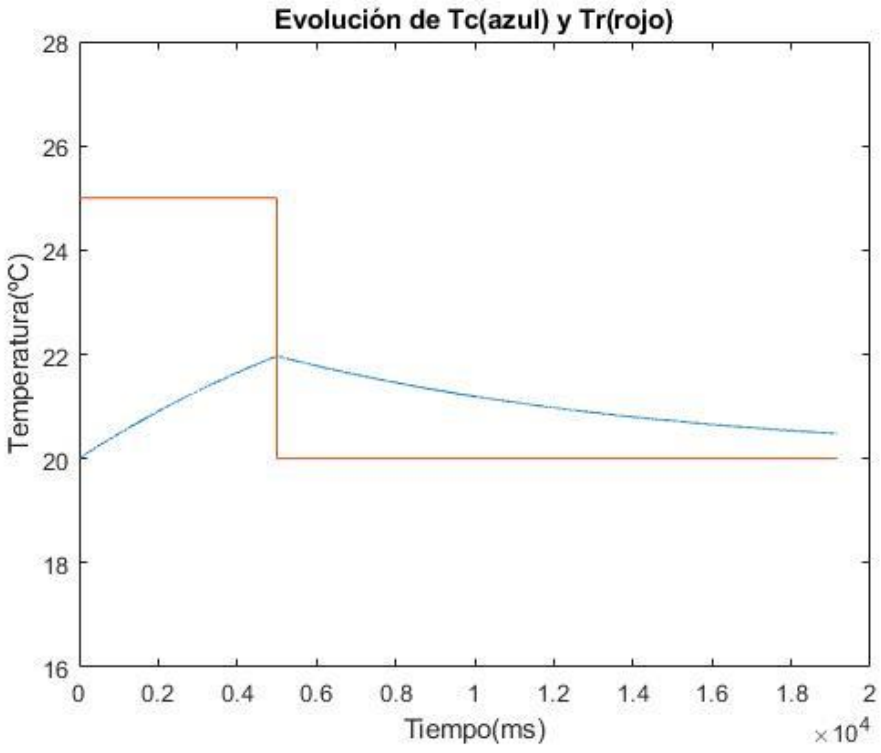

Figura 28. Gráfica de Tc y Tr para t=5000

Como ya he dicho antes, observamos como Tc, no llega a alcanzar el escalón, y esto es debido al poco tiempo de evolución hasta el escalón.

Después de realizar varias iteraciones, incrementando el valor de *t* en el escalón y Lj, conseguimos llegar a los valores deseados, para una $t = 43000$ ms y $Lj = 3500$ peticiones.

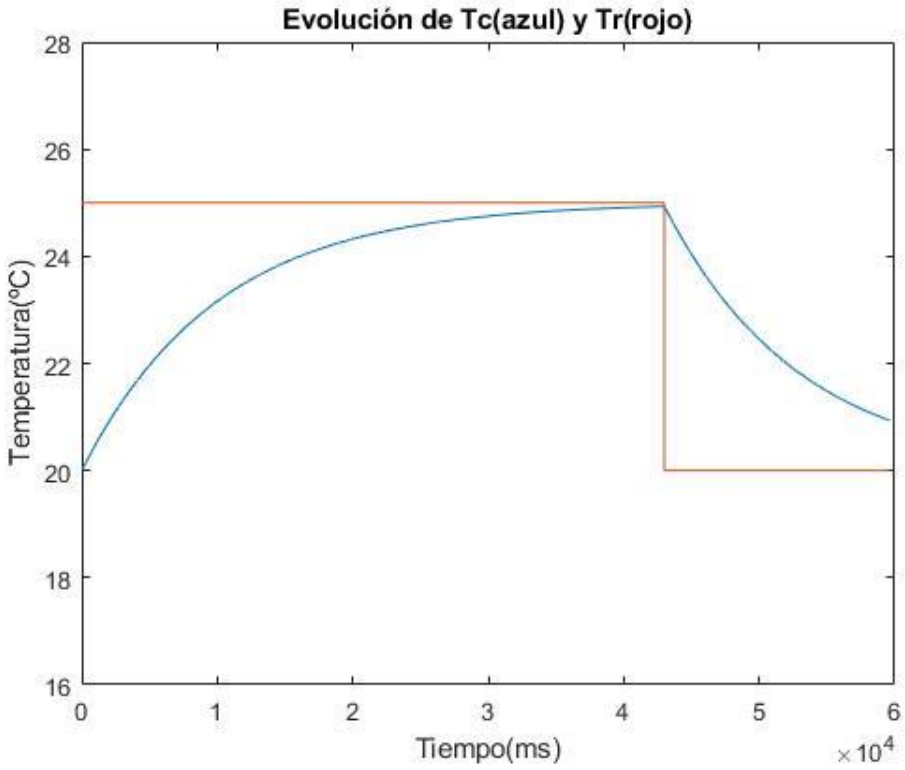

Figura 29. Gráfica Tc y Tr, ganancia alcanzada.

Como se puede observar solo era cuestión de tiempo que esto ocurriera, se ve perfectamente en la gráfica como la temperatura de la unidad aislada de frío alcanza la temperatura deconsigna del aire de salida de la máquina.

5.3 Respuesta del sistema ante un cambio brusco en el número de servidores

Ahora vamos a hacer la simulación para que de $t = 0$ a $t < 6000$ esten los 10 servidores, pero cuando $t \geq 6000$ dejaremos únicamente 3 servidores.

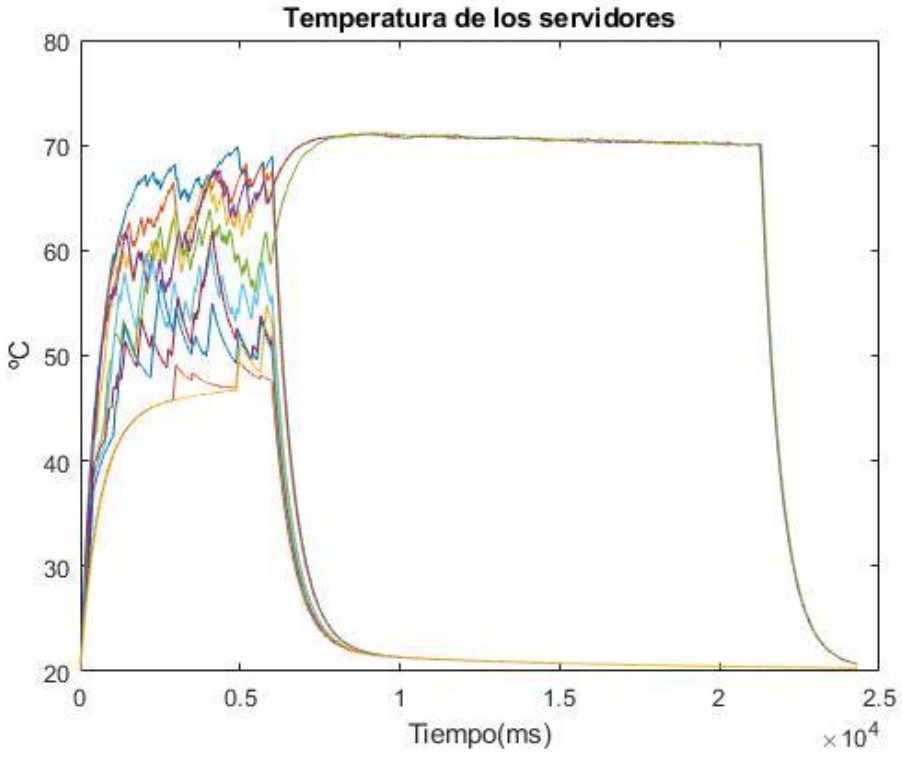

Figura 30. Gráfico de temperaturas para un cambio brusco

Al cambiar bruscamente los servidores operativos, se ve como todos van subiendo sus temperaturas desde 0 hasta $0.6x10^4$, a partir de ahí, para los para los 3 servidores que se quedan encendidos la temperatura asciende hasta 70ºC más o menos, donde se mantiene. Sin embargo, para los 7 servidores que se apagan la temperatura va reduciendo hasta llegar a sus 20ªC iniciales.

Ahora vamos a ver que pasa con las colas en este caso.

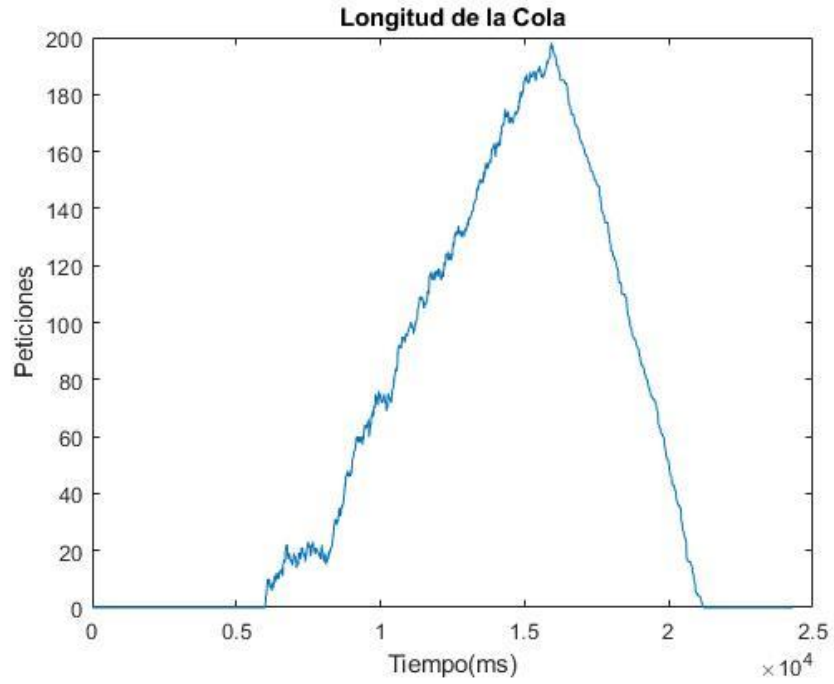

Figura 31. Longitud de cola para un cambio brusco

En este gráfico se puede apreciar perfectamente como en el intervalo de tiempo de $[0,\ \ 0.6x10^4]$ no se genera ningún tipo de cola ya que están trabajando los 10 servidores. Sin embargo, nada mas que se apagan los 7 servidores la cola empieza a aumentar muy rápidamente llegando a valores bastante altos, esta cola solo parará de aumentar si dejan de llegar peticiones, como en este caso que hay un límite.

Ahora vamos a fijarnos en la potencia

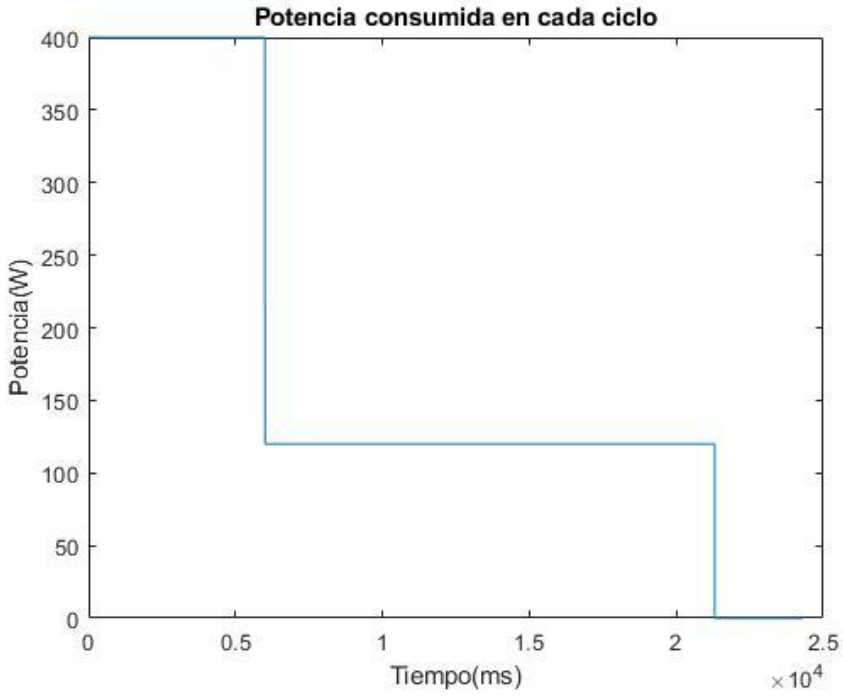

Figura 32. Potencia consumida para un cambio brusco

Este dato resulta curioso, porque al no tener en cuenta la frecuencia a la que se trabaja, que en este caso habría que aumentarla para así no llegar a tener grandes colas, la potencia consumida disminuye notablemente cuando se apagan los 7 servidores, ya que únicamente hay 3 consumiento y nuestras variables a1 y a2 son constantes.

5.4 Comparación del modelo de Matlab® con el Modelo de Arena®

Mi compañera y amiga Celia Regidor, a modelado el mismo modelo dinámico, pero en vez de usando el programa Matlab®, lo ha modelado con el simulador de eventos discretos Arena®.

Me gustaría hacer una breve comparación en base a las temperaturas de los servidores, que me parece un dato bastante sólido y relevante, para corroborar que los dos modelos funcionan correctamente y aportan resultados lógicos, para ello voy a volver a recurrir al primer escenario de simulación estudiado y a su correspondiente gráfica de temperatura, y esta la voy a comparar con el modelo de Arena en unas situaciones prácticamente iguales, para ver como se produce el desarrollo de los cambios, al modificar los servidores operativos, en las temperaturas.

Como puede apreciarse prácticamente a simple vista en las figuras 25 y 33, se ve que para unas situaciones muy similares se comportan de manera muy parecida, por que los dos modelos podrían ser perfectamente equivalentes.

Se puede ver como en el mismo punto, en el t = 1200ms o en el t = 1.2s (para el arena), como las temperaturas de todos los servidores menos de uno empiezan a bajar hasta alcanzar la temperatura mínima de 20ºC.

También se ve como ambas graficas tienen unos valores de temperatura límites superiores muy semejantes sin haber sido introducidos directamente en el programa, ya que la temperatura no es una variable que pueda manipularse.

Cuando parece que comienza a equilibrase, esas obscilaciones son debidas a la llegada aleaotria de peticiones u ordenes.

Matlab®

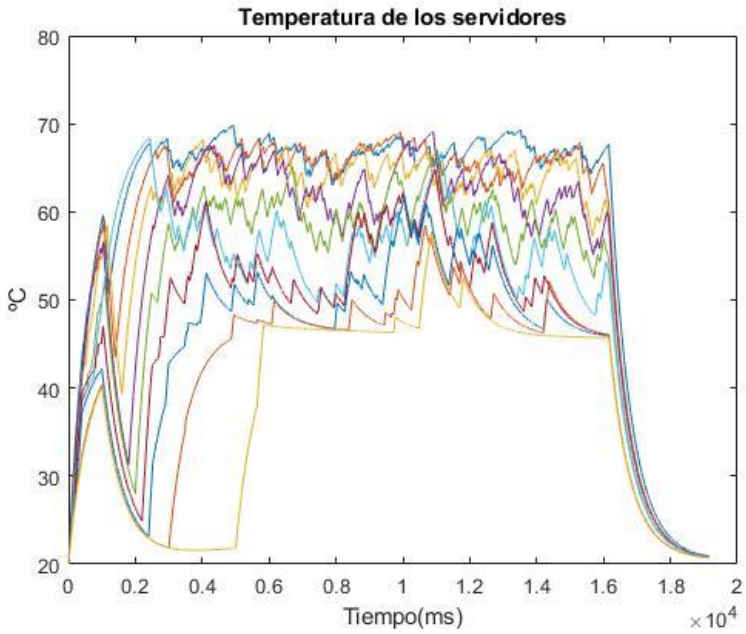

Figura 25. Gráfico de temperaturas de los servidores.

Arena®

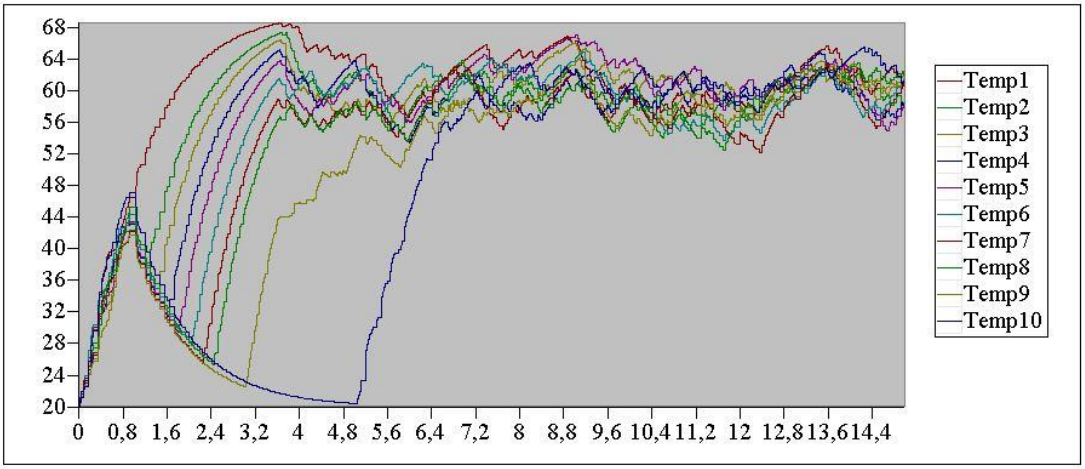

Figura 33. Gráfico de temperaturas de Arena.

6 CONCLUSIONES

Después de haber realizado el modelo de simulación con Matlab, se pueden sacar varias conclusiones a través de los datos obtenidos, a parte de una visualización del sistema mucho más clara.

Las conclusiones que podríamos sacar es que a la hora de disminuir los servidores activos aumenta la temperatura y viceversa, lo que conlleva a que un aumento en los servidores disminuiría la temperatura de los servidores y reduciría las colas, el problemas es que al aumentar estos servidores también se esta aunmentando el consumo de la potencia, por lo que hay que hacerlo de la manera adecuada, esto se podría realizar implantando unos controladores básicos que gestionaran la operatividad de los servidores, según alguna de las variables que hemos visto, con las que guardan bastante relación, como podría ser el tamaño de cola o las temperaturas, o incluso los dos.

El caso es que este tipo de modelos abren muchas puertas a la hora de la experimentación, a partir de aquí pueden probarse cantidad de situaciones diferentes para estudiar el comportamiento energético y la gestión de tareas de dichos CPD.

REFERENCIAS

John Spacey, Components of a Data Center, October 2017. [En línea]. Available: https://simplicable.com/new/data-centers

Ordenadores y portátiles, Problemas en los centros de procesamiento de datos [En línea]. Available: http://www.ordenadores-y-portatiles.com/centros-de-procesamiento-de-datos.html

Uptime Institute, 2009. Welcome to Green IT 2.0, the "Second Wave"- Now What Are You Going to Do About It. [En línea] Available: https://uptimeinstitute.com/

[1] Information and Communication Technology Office (ICTO), Conserving Energy, Hot and Cold Aisles in Data Center. [En línea] Available: https://newsletter.icto.um.edu.mo/conserving-energy-hot-and-cold-aisles-in-data-center/

[2] Above Infranet, Our Racks and Cooling Solution. [En línea]. Available: http://aboveinfranet.com/solutions/it-infrastructure-company/racks-cooling-solution/

Lijun Fu, Dynamic thermal and IT resource management strategies for data center energy minimization, 2017

[3] MathWorks®, Descripción del product MATLAB, [En línea] Available: https://es.mathworks.com/help/matlab/learn_matlab/product-description.html