

**Universidad Nacional del Nordeste**  
**Facultad de Ciencias Exactas y Naturales y**  
**Agrimensura**

**Monografía de Adscripción**

**Asignatura: Sistemas Operativos**

*Desarrollo de Applets Para  
la Simulación de  
Algoritmos de  
Administración del Sistema  
Operativo*

**APELLIDO Y NOMBRE:** *Rodríguez Nelson Fabián*

**L.U:** 32945

**PROF. DIRECTOR:** *David Luis La Red Martínez*

**LICENCIATURA EN SISTEMAS DE INFORMACIÓN**

*Corrientes - Argentina*

2009

# **Análisis del Rendimiento de un Subsistema de Disco de Una Petición a la Vez**

## **Introducción**

Este trabajo práctico de Adscripción consiste en la investigación, análisis y desarrollo de los diferentes casos de estudio propuestos en el SistOper, se han desarrollados applets utilizando el lenguaje **Java** y como IDE a NetBeans, una poderosa herramienta para el desarrollo de aplicaciones orientadas a objetos y el desarrollo de applets para su utilización en Internet.

Los casos de estudio desarrollados fueron: Análisis del Rendimiento de un Subsistema de Disco de Una Petición a la Vez,, Análisis del Rendimiento de un Subsistema de Disco de Varias Peticiones a la Vez; este informe consta de un marco teórico de cada caso de estudio, el código desarrollado y cada una de las ventanas de los applets.

## Objetivo del Caso de Estudio

El objetivo del presente caso consistió en la realización de un programa en JAVA que implementara el algoritmo de análisis de rendimiento para el caso señalado.

Asimismo también se estableció como objetivo del caso la inclusión en el programa de la posibilidad de generar información detallada respecto de los cálculos efectuados con las distintas simulaciones y un análisis estadístico de los resultados logrados, como así también un gráfico ilustrativo de los mismos.

## Descripción del problema planteado

Si existe una población de clientes que demandan cierto servicio prestado por servidores:

Algunos clientes ingresarán a la red de colas y esperarán que un servidor quede disponible.

Algunas colas son:

*Ilimitadas* pueden crecer tanto como sea necesario para contener a los clientes que esperan.

*Limitadas* solo pueden contener un número fijo de clientes en espera y quizás hasta ninguno.

Se deben tener en cuenta *variables aleatorias* que pueden ser descritas por *distribuciones probabilísticas*.

La variable aleatoria “q” representa el tiempo que emplea un cliente esperando en la cola a ser servido.

La variable aleatoria “s” representa la cantidad de tiempo que emplea un cliente en ser servido.

La variable aleatoria “w” representa el tiempo total que emplea un cliente en el sistema de colas: “w = q + s”.

### **Fuente.**

Los clientes son proporcionados a un sistema de colas desde una *fente* que puede ser infinita o finita.

Con una *fente infinita* la cola de servicio puede llegar a ser arbitrariamente grande.

Para una *fente finita* la cola de servicio es limitada.

Una fuente finita pero muy grande suele considerarse como infinita.

### **Llegadas.**

Supondremos que los clientes llegan a un sistema de colas

En los tiempos:

$$t_0 < t_1 < t_2 < \dots < t_n.$$

Los clientes llegan de uno en uno y nunca hay una colisión.

Las variables aleatorias “  $\tau_k$  ” miden los tiempos entre las llegadas sucesivas (arbitrario) y se denominan *tiempos entre*

*Llegadas:*

Son variables aleatorias *independientes* y están *Idénticamente distribuidas*.

$$\tau_k = t_k - t_{k-1} \quad (k \geq 1).$$

### **Llegadas de poisson.**

Las llegadas pueden seguir distintos patrones arbitrarios

Pero suele suponerse que forman un *proceso de llegadas de poisson*:

- Los tiempos entre llegadas están distribuidos exponencialmente:

$$P(\tau \leq t) = 1 - e^{-\lambda t}$$

- La probabilidad de que lleguen exactamente  $n$  clientes en cualquier intervalo de longitud  $t$  es:

$$[e^{-\lambda t} (\lambda t)^n] / n! \quad (n = 0, 1, 2, \dots).$$

- $\lambda$  es una tasa promedio de llegadas constante expresada en “clientes por unidad de tiempo”.
- El número de llegadas por unidad de tiempo se dice que tiene *distribución de poisson* con una media  $\lambda$ .

### **Tiempos de servicio.**

Se supone que los tiempos de servicio son aleatorios.

“ $s_k$ ” es el tiempo de servicio que el  $k$ -ésimo cliente requiere del Sistema.

Un tiempo de servicio arbitrario se designa por “ $s$ ”.

La distribución de los tiempos de servicio es:

- $W_s(t) = p(s \leq t)$ .

Para un servicio aleatorio con una tasa promedio de servicio “ $\mu$ ”:

- $W^s(t) = p(s \leq t) = 1 - e^{-\mu t} \quad (t \geq 0)$ .

### **Capacidad de la cola.**

Las colas deben tener:

*Capacidad infinita:*

- Cada cliente que llegue puede entrar en el sistema de colas y esperar, independientemente de cuantos clientes hay en espera.

*Capacidad cero (o sistemas de pérdidas):*

- Los clientes que llegan cuando la instalación de servicio está ocupada no podrán ser admitidos al sistema.

*Capacidad positiva:*

- Los clientes que llegan solo esperan si hay lugar en la cola.

### **Numero de servidores en el sistema.**

Los sistemas de colas se pueden categorizar según el número de Servidores en:

*Sistemas de un solo servidor:*

- Tienen un solo servidor y nada más pueden darle servicio a un solo cliente a la vez.

*Sistemas de servidores múltiples:*

- Tienen “c” servidores con idéntica capacidad y pueden dar servicio a “c” clientes a la vez.

## **RESUMEN DEL PROCESO DE POISSON**

Se define “p(k,t)” como la probabilidad de exactamente “k” llegadas en un intervalo de tiempo de longitud “t”.

Un proceso es de poisson si y solo si:

Para intervalos apropiadamente pequeños  $\Delta t$ :

- $P(k,t) = :$ 
  - $\lambda \Delta t$  para  $k = 1$  ( $\lambda$  es la tasa promedio de llegadas).
  - $1 - \lambda \Delta t$  para  $k = 0$ .
  - 0 para  $k > 1$ .

Cualesquiera eventos definidos para tener lugar en intervalos de tiempo no superpuestos son mutuamente independientes.

Un proceso también es de poisson si los tiempos entre llegadas sucesivas (*tiempos entre llegadas de primer orden*):

Son variables aleatorias exponenciales idénticamente distribuidas.

Si la variable aleatoria “k” indica el número de llegadas:

La *probabilidad* de, exactamente, “k” llegadas en un intervalo de longitud “t” es:

- $P(k,t) = [(\lambda t)^k e^{-\lambda t}] / k! \quad t \geq 0; k = 0, 1, 2, \dots$

El *valor esperado* o *valor medio* de k es:

- $E(k) = \lambda t$ .

La *varianza* de k es:

- $(\sigma_k)^2 = \lambda t$ .

La suma de dos variables de poisson aleatorias independientes “x” e “y” también describen un proceso de poisson:

Los valores esperados son:

- $E(y) = \mu_2 = \lambda_2 t.$
- $E(x) = \mu_1 = \lambda_1 t.$

La probabilidad de “k” llegadas en el tiempo “t” es:

$$\begin{aligned}
 P(k,t) &= [(\lambda_1 t + \lambda_2 t)^k e^{-(\lambda_1 t + \lambda_2 t)}] / k! & t \geq 0; \quad k = 0, 1, 2, \dots \\
 P(k,t) &= [(\mu_1 + \mu_2)^k e^{-(\mu_1 + \mu_2)}] / k! \\
 P(k,t) &= [\mu_s^k e^{-\mu_s}] / k! & \mu_s = \mu_1 + \mu_2 \\
 P(k,t) &= [(\lambda_s t)^k e^{-\lambda_s t}] / k! & \lambda_s = \lambda_1 + \lambda_2
 \end{aligned}$$

La suma de “n” procesos de poisson independientes resulta en un proceso de poisson con una tasa de llegada:

$$\lambda = \sum_{i=1}^n \lambda_i$$

Para un proceso de poisson con una tasa de llegada “ $\lambda$ ” se puede formar un nuevo proceso de poisson utilizando *borradas aleatorias independientes*:

Cada llegada al proceso original:

- Se acepta al nuevo proceso con probabilidad “p”.
- Se rechaza con probabilidad “1 - P”.

La tasa de llegada del nuevo proceso derivado es “ $\lambda P$ ”.

La generalización para la descomposición de un proceso de poisson en “n” procesos derivados independientes, cada uno con una probabilidad asociada “ $p_i$ ” resulta:

$$\lambda_n = p_n \lambda.$$

$$\sum_{i=1}^n p_i = 1.$$

$$\sum_{i=1}^n \lambda_i = \sum_{i=1}^n p_i \lambda = \lambda \sum_{i=1}^n p_i = \lambda.$$

En un proceso de poisson:

La probabilidad de que no haya llegadas en un intervalo de longitud “t” es:

- $P(0,t) = [(\lambda t)^0 e^{-\lambda t}] / 0! = e^{-\lambda t}.$

La probabilidad de una o más llegadas en un intervalo de longitud “t” es:

- $1 - P(0,t) = 1 - e^{-\lambda t}.$

La función de densidad de probabilidad para el *tiempo entre llegadas de primer orden (tiempo hasta la primera llegada)* es:

$$f_t(t) = \lambda e^{-\lambda t} \quad (t \geq 0).$$

El valor esperado “t” es:

- $E(t) = 1 / \lambda.$

La varianza es:

- $(\sigma_t)^2 = 1 / \lambda^2.$

La función de densidad de probabilidad para el *tiempo entre llegadas de orden r-esimo (tiempo hasta la r-esima llegada)* es:

- $f_t(t) = \lambda^r t^{r-1} e^{-\lambda t} / (r - 1)! \quad (t \geq 0, r = 1, 2, \dots)$

El valor esperado “t” es:

- $E(t) = r / \lambda.$

La desviación estándar es:

- $(\sigma_t)^2 = r / \lambda^2.$

Las instalaciones de servicio pueden proporcionar tiempos de servicio exponenciales:

La probabilidad de que el tiempo de servicio sea menor o igual a “t” es:

- $P(S \leq t) = 1 - e^{-\mu t} \quad (t \geq 0).$

La tasa promedio de servicio es “ $\mu$ ”.

El tiempo promedio de servicio es “ $1 / \mu$ ”.

La función de densidad de probabilidad para el tiempo de servicio “t” es:

- $f_t(t) = \mu e^{-\mu t} \quad (t \geq 0).$

La media del tiempo de servicio es:

- $E(s) = 1 / \mu.$

La varianza es “ $1 / \mu^2$ ”.

Un servidor que opera de esta manera se denomina *servidor exponencial*.

## **Descripción del algoritmo desarrollado**

Se supone la siguiente situación:

Las peticiones de acceso a disco llegan como un proceso de poisson con una tasa promedio de  $\lambda$  peticiones por minuto.

Si el disco está en uso, la petición se coloca en una cola primero en llegar, primero en ser servido.

Cuando el disco queda disponible se sirve la primera petición de la cola.

El tiempo de servicio es una variable aleatoria exponencialmente distribuida con un valor esperado de  $1 / \mu$  minutos.

La tasa promedio de servicio es de  $\mu$  peticiones por minuto.

Se debe determinar, para c / u de los casos:

- El valor esperado para el número total de peticiones al disco pendientes (en la cola o en servicio).
- Las probabilidades del estado limite.
- En este caso el dispositivo de disco contiene un solo brazo.
- Solo puede dar servicio a una petición a la vez.
- La tasa de servicio es  $\mu$ .

## Solución propuesta

$S_i$  es el estado del sistema cuando hay  $i$  peticiones de disco al dispositivo de servicio de disco.

La tasa de llegadas de peticiones es independiente del estado del sistema:

- La probabilidad de la transición  $s_i \rightarrow s_{i+1}$  en el siguiente

Intervalo de tiempo  $\Delta t$  es  $\lambda \Delta t$ .

Se considera al sistema como un proceso de nacimiento y muerte continuo de cadena sencilla y estados infinitos con:

- $d_i = 0 \quad i = 0.$
- $d_i = \mu \quad i = 1, 2, 3, \dots$
- $b_i = \lambda \quad i = 0, 1, 2, \dots$

Solo una petición puede ser servida en un momento dado y se sirve a una tasa  $\mu$ .

$\mu > \lambda$ :

- Asegura que la longitud de la cola de peticiones en espera no crezca indefinidamente.

Se utilizan las relaciones:

$$P_{i+1} = (b_i / d_{i+1}) P_i \quad i = 0, 1, 2, \dots$$

$$\sum_i P_i = 1.$$

$$P_1 = (\lambda / \mu) P_0.$$

$$P_2 = (\lambda / \mu) P_1 = (\lambda / \mu)^2 P_0.$$

$$P_i = (\lambda / \mu)^i P_0.$$

$$\sum_i P_i = 1 = \sum_i (\lambda / \mu)^i P_0 = 1 / [1 - (\lambda / \mu)] P_0.$$

$P_0 = 1 - (\lambda / \mu)$ : probabilidad de que el sistema se encuentre ocioso.

$$P_i = (\lambda / \mu)^i P_0 = [1 - (\lambda / \mu)] (\lambda / \mu)^i, \quad i = 0, 1, 2, \dots$$

$P_i = [1 - (\lambda / \mu)] (\lambda / \mu)^i$  : probabilidad que hayan  $i$  peticiones pendientes.

El número promedio de peticiones pendientes es:

$$E(i) = \sum_i i P_i = \sum_i i [1 - (\lambda / \mu)] (\lambda / \mu)^i = [1 - (\lambda / \mu)] \sum_i i (\lambda / \mu)^i =$$

$$E(i) = [1 - (\lambda / \mu)] (\lambda / \mu) \sum_i i (\lambda / \mu)^{i-1} =$$

$$E(i) = [1 - (\lambda / \mu)] (\lambda / \mu) \{1 / [1 - (\lambda / \mu)^2]\} =$$

$$E(i) = (\lambda / \mu) \lambda / \mu [1 - (\lambda / \mu) \lambda / \mu]^{-1}.$$

## Applet desarrollado

```
/*
(* TEORIA DE COLAS *)
(* Análisis del rendimiento de un subsistema de disco *)
(* Caso 1: El subsistema de disco solo puede dar servicio a una petición a la vez. *)
(* Referencias y aclaraciones:
mu: Tasa promedio de servicio para un servidor.
mu = 1/Es(s)
Es(s): Tiempo de espera de servicio para un cliente.
lambda: Tasa promedio de llegadas de clientes al sistema de colas.
lambda = 1/Es(tau)
Es(tau): Tiempo de espera entre llegadas.
Es(tau) = 1/lambda
mu > lambda: Restricción para que la longitud de la cola de peticiones no crezca indefinidamente.
tau: Intervalo entre llegadas.
Pn: Probabilidad de que haya "n" clientes en el sistema de colas en estado estable.
P0: Probabilidad de que el sistema se encuentre ocioso.
P0 = 1-(lambda/mu)
Pri: Probabilidad de que hayan "i" peticiones pendientes.
Pri = (1-(lambda/mu))(lambda/mu)^i
(i=0,1,2,3,..)
Es(i): N° promedio de peticiones pendientes.
Es(i) = (lambda/mu)(1-(lambda/mu))^(i-1) *)
*/
package subsistdiscounapet;
public class Analisis {
    double tau,Pn,P0,muG,lambdaG,petPenG;
    private java.util.ArrayList<Double> listaCoef;
    private java.util.ArrayList<Double> listaPetPend;
    private java.util.ArrayList<double[]> listaPuntos;
    public Analisis(double mu,double lambda, int petPen){
        this.listaCoef=new java.util.ArrayList();
        this.listaPetPend= new java.util.ArrayList();
    }
}
```

```

this.listaPuntos= new java.util.ArrayList<double[]>();
muG=mu; lambdaG=lambda; petPenG=petPen;
}
public String salida(){
this.getListCoef().clear();
this.getListPetPend().clear();
this.getListPuntos().clear();
String aux= "Análisis del rendimiento de un subsistema \n"+
"de disco que solo puede dar servicio a una "+
"petición a la vez.\n"+
"Colocar los valores máximos para mu, lambda \ny el número de peticiones
pendientes.\n*****\n\n";
for(int i=10;i<=muG;i+=10)
for(int j=1; j<lambdaG;j+=2)
aux+= caso1(i,j,3);
aux+="\n\n*****";
aux+="\n***** RESUMEN FINAL *****";
aux+="\n*****";
aux+="\n\n Lista de Coeficientes Mu/Lambda:\n{";
for(int i=0;i<this.listaCoef.size();i++){
if(i%10==0 && i!=0) aux+="\n";
if(i!=(this.listaCoef.size()-1))
aux+=" "+((double)this.listaCoef.get(i))+";";
else
aux+=" "+((double)this.listaCoef.get(i))+"}";
}
aux+="\n\n Lista de Peticiones Pendientes:\n{";
for(int i=0;i<this.listaPetPend.size();i++){
if(i%10==0 && i!=0) aux+="\n";
if(i!=(this.listaPetPend.size()-1))
aux+=" "+((double)this.listaPetPend.get(i))+";";
else
aux+=" "+((double)this.listaPetPend.get(i))+"}";
}
aux+="\n\n Lista de Pares Ordenados:\n{";
for(int i=0;i<this.listaPuntos.size();i++){
if(i%10==0 && i!=0) aux+="\n";
if(i!=(this.listaPuntos.size()-1))
aux+=" (" +this.listaPuntos.get(i)[0]+", "+this.listaPuntos.get(i)[1]+");";
else
aux+=" (" +this.listaPuntos.get(i)[0]+", "+this.listaPuntos.get(i)[1]+");";
}
}
return aux;
}
private double pri(double mu,double lambda,int i){
return ((1-(lambda*Math.pow(mu,-1))) * Math.pow((lambda*Math.pow(mu,-1)),i));
}
private double es(double mu,double lambda){
return(lambda*Math.pow(mu,-1))* Math.pow((1-(lambda*Math.pow(mu,-1))),(i));
}
private String caso1(double mu,double lambda,int i){
String salida="";
double coef;
if((lambda/mu)<1){
salida+="Para que la cola no crezca indefinidamente\n debe ser mayor MU que LAMBDA.\n";
coef=(lambda/mu);//TODO
salida+="Analisis de rendimiento de un subsistema de disco.\n";
salida+="Resultado del Caso1:\n";
salida+="El subsistema de disco solo puede dar\n servicio a una peticion a la vez.\n";
salida+="Los valores de mu, lambda y el coeficiente\n son los siguientes:\n";
salida+=""+mu+", "+lambda+", "+coef+"\n";
salida+="La cola no crecera eindefinidamente debido\n a que Mu es mayor que Lambda.\n";
}
}

```

```

        salida+="La probabilidad de tener 0,1,2,3,4,... peticiones\n";
        salida+="pendientes son las siguientes: {"";
        for(int j=0;j<=i;j++)
            if(j!=i)
                salida+=""+pri(mu,lambda,j)+" ,";
            else
                salida+=""+pri(mu,lambda,j);
        salida +="}\n";
        // salida+="pendientes son las siguientes:"+ pri(mu,lambda,i) +"\n";
        salida+="El promedio de peticiones pendientes\n es el siguiente: "+es(mu,lambda) +"\n";
        this.getListaCoef().add(coef);
        this.getListaPetPend().add(es(mu,lambda));
        double[] punto=new double[2];
        punto[0]=coef;
        punto[1]=es(mu,lambda);
        this.getListaPuntos().add(punto);
        salida+="*****\n";
    }
    return salida;
}

/**
 * @return the listaCoef
 */
public java.util.ArrayList getListaCoef() {
    return listaCoef;
}

/**
 * @return the listaPetPend
 */
public java.util.ArrayList getListaPetPend() {
    return listaPetPend;
}

/**
 * @return the listaPuntos
 */
public java.util.ArrayList<double[]> getListaPuntos() {
    return listaPuntos;
}

}

package subsistdiscounapet;

/**
 *
 * @author
 */
public class AnalisisDeUnaPeticion extends javax.swing.JApplet {

    /** Initializes the applet AnalisisDeUnaPeticion */
    public void init() {
        try {
            java.awt.EventQueue.invokeAndWait(new Runnable() {
                public void run() {
                    initComponents();
                }
            });
        } catch (InterruptedException ex) {
            Logger.getLogger(AnalisisDeUnaPeticion.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

```

```

    }
  });
} catch (Exception ex) {
    ex.printStackTrace();
}
}

/** This method is called from within the init() method to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jDialog1 = new javax.swing.JDialog();
    jLabel4 = new javax.swing.JLabel();
    jButton2 = new javax.swing.JButton();
    jScrollPane1 = new javax.swing.JScrollPane();
    jTextArea1 = new javax.swing.JTextArea();
    jLabel1 = new javax.swing.JLabel();
    jTextField1 = new javax.swing.JTextField();
    jLabel2 = new javax.swing.JLabel();
    jTextField2 = new javax.swing.JTextField();
    jTextField3 = new javax.swing.JTextField();
    jLabel3 = new javax.swing.JLabel();
    jPanel1 = new javax.swing.JPanel();
    jButton1 = new javax.swing.JButton();
    jLabel5 = new javax.swing.JLabel();
    jLabel6 = new javax.swing.JLabel();

    jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    jLabel4.setText("jLabel4");

    jButton2.setText("Aceptar");
    jButton2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton2ActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout jDialog1Layout = new javax.swing.GroupLayout(jDialog1.getContentPane());
    jDialog1.getContentPane().setLayout(jDialog1Layout);
    jDialog1Layout.setHorizontalGroup(
        jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jDialog1Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 272,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton2))
                .addContainerGap(134, true)))
    );
    jDialog1Layout.setVerticalGroup(
        jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jDialog1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel4)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton2)
                .addContainerGap(134, true)))
    );
}

```

```

        .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 50,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jButton2)
        .addContainerGap(32, Short.MAX_VALUE))
    );

    jTextArea1.setColumns(20);
    jTextArea1.setEditable(false);
    jTextArea1.setRows(5);
    jTextArea1.setWrapStyleWord(true);
    jScrollPane1.setViewportView(jTextArea1);

    jLabel1.setFont(new java.awt.Font("Tahoma", 1, 14));
    jLabel1.setText("Mu");

    jTextField1.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
    jTextField1.setText("40");

    jLabel2.setFont(new java.awt.Font("Tahoma", 1, 14));
    jLabel2.setText("Lambda");

    jTextField2.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
    jTextField2.setText("20");

    jTextField3.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
    jTextField3.setText("3");

    jLabel3.setFont(new java.awt.Font("Tahoma", 1, 14));
    jLabel3.setText("Peticiones \nPendientes");

    jPanel1.setBackground(new java.awt.Color(255, 255, 255));
    jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

    javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
    jPanel1.setLayout(jPanel1Layout);
    jPanel1Layout.setHorizontalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(
                jPanel1Layout.createSequentialGroup()
                    .addGap(0, 497, Short.MAX_VALUE)
            )
    );
    jPanel1Layout.setVerticalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(
                jPanel1Layout.createSequentialGroup()
                    .addGap(0, 217, Short.MAX_VALUE)
            )
    );

    jButton1.setFont(new java.awt.Font("Tahoma", 1, 12));
    jButton1.setText("Calcular");
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
        }
    });

    jLabel5.setFont(new java.awt.Font("Tahoma", 1, 14));
    jLabel5.setText("Detalles de los calculos realizados");

    jLabel6.setFont(new java.awt.Font("Tahoma", 1, 14));
    jLabel6.setText("Gráfico Estadístico");

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(

```

```

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addContainerGap()
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 418,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jLabel6))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jLabel2)
.addComponent(jLabel1)
.addComponent(jLabel3))
.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 83,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addGap(21, 21, 21)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
.addComponent(jTextField3)
.addComponent(jTextField1)
.addComponent(jTextField2, javax.swing.GroupLayout.DEFAULT_SIZE, 73, Short.MAX_VALUE)))
.addComponent(jLabel5)
.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
);
layout.setVerticalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
.addGroup(layout.createSequentialGroup()
.addGap(17, 17, 17)
.addComponent(jLabel5)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 207,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addGroup(layout.createSequentialGroup()
.addGap(57, 57, 57)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
.addGroup(layout.createSequentialGroup()
.addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(67, 67, 67))
.addGroup(layout.createSequentialGroup()
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jLabel1)
.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
.addGap(28, 28, 28)
.addComponent(jLabel2)
.addGap(37, 37, 37)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 33,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 31,
javax.swing.GroupLayout.PREFERRED_SIZE))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

```

```

        .addComponent(jLabel6)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(33, 33, 33)
    );
} // </editor-fold>

```

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    double mu, lambda;
    int i;
    try {
        mu=Integer.parseInt(this.jTextField1.getText());
        lambda=Integer.parseInt(this.jTextField2.getText());
        if(mu>lambda){
            i=Integer.parseInt(this.jTextField3.getText());
            Analisis a = new Analisis(mu,lambda,i);
            this.jTextArea1.setText(a.salida());
            this.listaValores=a.getListaPuntos();
            dibujar();
        } else {
            this.jLabel4.setText("Mu debe ser mayor a Lambda");
            this.jDialog1.setSize(300,180);
            this.jDialog1.setLocation(300,200);
            this.jDialog1.setVisible(true);
        }
    } catch(Exception e){
        // javax.swing. // dia=new javax.swing.JDialog();
        this.jLabel4.setText("Error en los parametros");
        this.jDialog1.setSize(300,180);
        this.jDialog1.setLocation(300,200);
        this.jDialog1.setVisible(true);
    }
}
}

```

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.jDialog1.hide();
}
private void dibujar() {

```

```

    double XMax=-10000;
    double YMax=-10000;
    for(int i=0;i<this.listaValores.size();i++){
        if(XMax < this.listaValores.get(i)[0])
            XMax=this.listaValores.get(i)[0];
        if(YMax < this.listaValores.get(i)[1])
            YMax=this.listaValores.get(i)[1];
    }
    java.awt.Graphics g = jPanel1.getGraphics();
    java.awt.Dimension d = jPanel1.getSize();
    apAncho = d.width;
    apAlto = d.height;
    g.setColor(java.awt.Color.WHITE);
    g.fillRect(0, 0, apAncho, apAlto);
    g.setColor(java.awt.Color.BLACK);
    g.drawRect(0, 0, apAncho-1, apAlto-1);
    xmin = -0.005;
    xmax = (XMax+0.03);
    ymin = -1.5;

```

```

        ymax = (YMax+0.03);
        int apCero=(int)((0-xmin) * (apAncho-1) / (xmax-xmin) );
        int ypCero= (int)((-0.3-ymin) * (apAlto-1) / (ymax-ymin) );
        ypCero = apAlto - ypCero;
        g.setColor(java.awt.Color.black);
        g.drawLine(0,ypCero,(int)apAncho,ypCero);
        g.drawLine(apCero,apAlto,apCero,0);
        int xp,yp;
        //yp = (int)((0.5-ymin) * (apAlto-1) / (ymax-ymin) );
        xp=(int)((0.5 - xmin) * (apAncho-1) / (xmax-xmin) );
        g.drawLine(apCero, ypCero, xp, ypCero);

        for(int i=0;i<this.listaValores.size();i++){
            double XP;
            XP=this.listaValores.get(i)[0];
            xp=(int)((XP - xmin) * (apAncho-1) / (xmax-xmin) );
            yp=(int)ValorY1(i);
            g.setColor(java.awt.Color.orange);
            g.fillOval(xp, yp, 7, 7);

        }
        double XP=0;
        while(XP < XMax){
            xp=(int)((XP - xmin) * (apAncho-1) / (xmax-xmin) );
            g.setColor(java.awt.Color.BLUE);
            String s="" +XP;
            s=s.substring(0,3);
            g.drawString(s,(int)xp,(int)ypCero+10);
            XP +=0.2;
        }
        int YP=0;
        while(YP < YMax){

            g.setColor(java.awt.Color.BLUE);
            String s="" +YP;

            g.drawString(s,(int)apCero,(int)ValorY2(YP));
            YP +=1;
        }
    }
    private int ValorY1(int valor) {
        double x, y;
        int retorno;
        x = (valor * (xmax - xmin) / (apAncho - 1)) + xmin;
        y=this.listaValores.get(valor)[1];
        retorno = (int)((y - ymin) * (apAlto - 1) / (ymax - ymin));
        retorno = apAlto - retorno;
        return (retorno);
    }
    private int ValorY2(double valor) {
        double y;
        int retorno;
        y=valor;
        retorno = (int)((y - ymin) * (apAlto - 1) / (ymax - ymin));
        retorno = apAlto - retorno;
        return (retorno);
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;

```

```

private javax.swing.JDialog jDialog1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JPanel jPanel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTextArea jTextArea1;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField2;
private javax.swing.JTextField jTextField3;
// End of variables declaration
private int apAlto,apAncho;
private double xmax,xmin,ymax,ymin;
private java.util.ArrayList<double[]> listaValores;
}

```

## Datos y ejecuciones

Al ejecutar el applet, comenzamos con unos datos predefinidos de  $\mu$  ( $\mu$ ),  $\lambda$  ( $\lambda$ ) y las peticiones pendientes, estos datos pueden ser modificados por el usuario siempre que sean válidos de acuerdo a la teoría de colas.

También podremos ver los resultados obtenidos en cada paso de la simulación en el panel superior de la ventana principal y un gráfico estadístico en el panel inferior de la misma ventana.

The screenshot displays the 'Ventana Principal' interface. At the top, there is a section titled 'Detalles de los calculos realizados' which contains a large empty rectangular area. To the right of this area are three input fields: 'Mu' with the value 40, 'Lambda' with the value 20, and 'Peticiones Pendientes' with the value 3. Below these fields is a blue 'Calcular' button. Underneath the 'Calcular' button is another section titled 'Gráfico Estadístico' which contains a large empty rectangular area. At the bottom left of this area is the label 'Esperas' and at the bottom center is the label 'Coeficientes'.

Ventana Principal

Panel de detalles de los pasos

**Detalles de los calculos realizados**

Lista de Peticiones Pendientes:  
{ 0.11111111111111112; 0.42857142857142866; 1.0; 2.3333333333333333  
1.2222222222222223; 1.8571428571428572; 3.0; 5.6666666666666670;  
0.5789473684210525; 0.7647058823529412; 1.0; 1.3076923076923076;  
0.37931034482758624; 0.48148148148148145; 0.6000000000000001;

Lista de Pares Ordenados:  
{ (0.1, 0.11111111111111112); (0.3, 0.42857142857142866); (0.5, 1.0); (0.  
(0.55, 1.2222222222222223); (0.65, 1.8571428571428572); (0.75, 3.0); (0.  
(0.36666666666666664, 0.5789473684210525); (0.43333333333333333;  
(0.275, 0.37931034482758624); (0.325, 0.48148148148148145); (0.375,

**Valores para la simulación**

Mu

Lambda

Peticiones Pendientes

**Calcular**

**Gráfico Estadístico**

Esperas

Coeficientes

Valores para la simulación

Botón para el inicio de la simulación

Gráfico estadístico

## **Resultados y conclusiones**

Los resultados obtenidos con el applet desarrollado son similares a los del programa desarrollado para el SistOper como era de esperarse, además, se verificó lo visto en teoría en cuanto a las diferencias en tiempos en cola de espera en disco de las distintas peticiones, según las distintas cargas de trabajo simuladas.

La modalidad implementada de mostrar los resultados paso a paso por pantalla permite observar el comportamiento del algoritmo y facilita la comprensión de su funcionamiento.

Asimismo se observa un fuerte impacto en los tiempos de espera a partir de cierto nivel de carga de trabajo, expresada según los coeficientes considerados.

# **Análisis del Rendimiento de un Subsistema de Disco de Varias Peticiones a la Vez**

## **Introducción**

Como los casos de estudio anteriores, este caso fue desarrollado en un applet para que pueda ser utilizado en Internet, en este caso software se desarrollo para la realización de un algoritmo que implemente el análisis de rendimiento de un subsistema de disco de varias peticiones a la vez, utilizando Teoría de Colas.

El estudio de las simulaciones efectuadas con el algoritmo señalado permite evaluar el comportamiento esperado de un subsistema de disco de varias peticiones a la vez cuando es sometido a distintas cargas de trabajo, es decir a conjuntos de peticiones de operaciones de acceso a los discos que configuran cantidades de trabajo distintas a ser atendidas.

La posibilidad de atender varias peticiones en paralelo requiere que el subsistema de disco disponga de un número elevado de brazos de acceso, que además sean independientes en sus movimientos, de tal manera que cada uno de ellos pueda atender a una petición.

## **Objetivo del Caso de Estudio**

Como en el algoritmo anterior, el objetivo del presente caso consiste en la realización de un programa en JAVA que implemente el algoritmo de análisis de rendimiento para el caso señalado.

Asimismo también se estableció como objetivo del caso la inclusión en el programa de la posibilidad de generar información detallada respecto de los cálculos efectuados con las distintas simulaciones y un análisis estadístico de los resultados logrados, como así también un gráfico ilustrativo de los mismos.

## **Descripción del problema planteado**

El problema que se plantea en este caso es el mismo que se desarrollo en el caso de estudio anterior de Análisis del Rendimiento de un Subsistema de Disco de Una Petición a la Vez

## Descripción del algoritmo desarrollado

Se supone la siguiente situación:

Las peticiones de acceso a disco llegan como un proceso de poisson con una tasa promedio de  $\lambda$  peticiones por minuto.

Si el disco está en uso, la petición se coloca en una cola primero en llegar, primero en ser servido.

Cuando el disco queda disponible se sirve la primera petición de la cola.

El tiempo de servicio es una variable aleatoria exponencialmente distribuida con un valor esperado de  $1 / \mu$  minutos.

La tasa promedio de servicio es de  $\mu$  peticiones por minuto.

Se debe determinar, para c / u de los casos:

- El valor esperado para el número total de peticiones al disco pendientes (en la cola o en servicio).
- Las probabilidades del estado limite.
- El dispositivo de disco contiene gran número de brazos móviles.
- Cada brazo puede dar servicio a una petición de disco a la misma tasa  $\mu$ .
- Se supone que un número infinito de peticiones pueden recibir servicio en paralelo.

Con  $i$  peticiones siendo servidas:

La probabilidad de que una petición en particular acabe siendo servida dentro del siguiente  $\Delta t$  es  $\mu \Delta t$ .

La probabilidad de que exactamente una petición cualquiera acabe es  $i \mu \Delta t$  (buena aproximación de primer orden).

Cualquiera de las  $i$  peticiones puede terminar y provocar un cambio de estado.

El sistema se ve como un proceso de nacimiento y muerte continuo de cadena sencilla y de estados infinitos con:

- $b_i = \lambda$        $i = 0, 1, 2, \dots$

- $d_i = 0 \quad i = 0.$
- $d_i = i\mu \quad i = 1, 2, 3, \dots$

Ningún cliente tiene que esperar ya que se suponen infinitos servidores en paralelo.

Se utilizan las relaciones:

$$P_{i+1} = (b_i / d_{i+1}) P_i. \quad i = 0, 1, 2, \dots$$

$$\sum_i P_i = 1.$$

$$P_1 = (\lambda / \mu) P_0.$$

$$P_2 = (\lambda / 2\mu) P_1 = (1 / 2) (\lambda / \mu)^2 P_0.$$

$$P_3 = (\lambda / 3\mu) P_2 = (1 / (3 \cdot 2)) (\lambda / \mu)^3 P_0.$$

$$P_i = (1 / i!) (\lambda / \mu)^i P_0.$$

$$\sum_i P_i = 1 = \sum_i (1 / i!) (\lambda / \mu)^i P_0.$$

$$\sum_n (x^n / n!) = e^x.$$

$$\sum_i P_i = 1 = \sum_i (1 / i!) (\lambda / \mu)^i P_0 = e^{\lambda / \mu} P_0.$$

$$P_0 = e^{-\lambda / \mu}.$$

$$P_i = (\lambda / \mu)^i [(e^{-\lambda / \mu}) / i!].$$

$$E(i) = \sum_i i P_i = \sum_i i (\lambda / \mu)^i [(e^{-\lambda / \mu}) / i!] = (e^{-\lambda / \mu}) \sum_i i (\lambda / \mu)^i (1 / i!) =$$

$$E(i) = (e^{-\lambda / \mu}) \sum_i (\lambda / \mu) (\lambda / \mu)^{i-1} [1 / (i-1)!] =$$

$$E(i) = (e^{-\lambda / \mu}) (\lambda / \mu) \sum_i [1 / (i-1)!] (\lambda / \mu)^{i-1} =$$

$$E(i) = (e^{-\lambda / \mu}) (\lambda / \mu) (e^{\lambda / \mu}) =$$

$$E(i) = (\lambda / \mu).$$

## Applet desarrollado

```
package subsistdiscovariaspet;
```

```
public class Analisis {
```

```
    double tau,Pn,P0,muG,lambdaG,petPenG;
```

```
    private java.util.ArrayList<Double> listaCoef;
```

```
    private java.util.ArrayList<Double> listaPetPend;
```

```

private java.util.ArrayList<double[]> listaPuntos;

public Analisis(double mu,double lambda, int petPen){

    this.listaCoef=new java.util.ArrayList();

    this.listaPetPend= new java.util.ArrayList();

    this.listaPuntos= new java.util.ArrayList<double[]>();

    muG=mu; lambdaG=lambda; petPenG=petPen;

}

public String salida(){

    this.getListaCoef().clear();

    this.getListaPetPend().clear();

    this.getListaPuntos().clear();

    String aux= "Análisis del rendimiento de un subsistema \n"+

        "de disco que puede dar servicio a varias "+

        "peticiones a la vez.\n"+

        "Colocar los valores máximos para mu, lambda \ny el número de peticiones
pendientes.\n*****\n\n";

    for(int i=10;i<=muG;i+=10)

        for(int j=1; j<lambdaG;j+=2)

            aux+= caso2(i,j,3);

    aux+="\n\n*****";

    aux+="\n***** RESUMEN FINAL *****";

    aux+="\n*****";

    aux+="\n\n Lista de Coeficientes Mu/Lambda:\n{";

    for(int i=0;i<this.listaCoef.size();i++){

        if(i%10==0 && i!=0) aux+="\n";

        if(i!=(this.listaCoef.size()-1))

            aux+=" "+((double)this.listaCoef.get(i))+",";

        else

            aux+=" "+((double)this.listaCoef.get(i))+"}";

    }

    aux+="\n\n Lista de Peticiones Pendientes:\n{";

    for(int i=0;i<this.listaPetPend.size();i++){

```

```

        if(i%10==0 && i!=0) aux+="\n";

        if(i!=(this.listaPetPend.size()-1))

            aux+=" "+((double)this.listaPetPend.get(i))+";";

        else

            aux+=" "+((double)this.listaPetPend.get(i)+"}";

    }

    aux+="\n\n Lista de Pares Ordenados:\n{";

    for(int i=0;i<this.listaPuntos.size();i++){

        if(i%10==0 && i!=0) aux+="\n";

        if(i!=(this.listaPuntos.size()-1))

            aux+=" (" +this.listaPuntos.get(i)[0]+", "+this.listaPuntos.get(i)[1]+");";

        else

            aux+=" (" +this.listaPuntos.get(i)[0]+", "+this.listaPuntos.get(i)[1]+"});";

    }

    return aux;

}

private double pri(double mu,double lambda,int i){

    return (Math.pow(lambda*Math.pow(mu, -1),i)*((Math.pow(Math.E,(lambda*Math.pow(mu, -1))))*(Math.pow(factorial(i),-1))));

        //(((1-(lambda*Math.pow(mu,-1))) * Math.pow((lambda*Math.pow(mu,-1)),i));

}

private int factorial(int i){

    int aux=1;

    for (int j=1;j<=i;j++)

        aux=aux*j;

    return aux;

}

private double es(double mu,double lambda){

    return(lambda*Math.pow(mu,-1));

}

private String caso2(double mu,double lambda,int i){

    String salida="";

```

```

double coef;
if((lambda/mu)<1){
    salida+="Para que la cola no crezca indefinidamente\n debe ser mayor MU que LAMBDA.\n";
    coef=(lambda/mu);//TODO
    salida+="Análisis de rendimiento de un subsistema de disco.\n";
    salida+="Resultado del Caso2:\n";
    salida+="El subsistema de disco puede dar\n servicio a varias peticiones a la vez.\n";
    salida+="Los valores de mu, lambda y el coeficiente\n son los siguientes:\n";
    salida+="""+mu+", "+lambda+", "+coef+"\n";
    salida+="La cola no crecera eindefinidamente debido\n a que Mu es mayor que Lambda.\n";
    salida+="La probabilidad de tener 0,1,2,3,4,... peticiones\n";
    salida+="pendientes son las siguientes: {";
    for(int j=0;j<=i;j++)
        if(j!=i)
            salida+="""+pri(mu,lambda,j)+" ";
        else
            salida+="""+pri(mu,lambda,j);
    salida +="}\n";
    // salida+="pendientes son las siguientes:"+ pri(mu,lambda,i) +"\n";
    salida+="El promedio de peticiones pendintes\n es el siguiente: "+es(mu,lambda) +"\n";
    this.getListaCoef().add(coef);
    this.getListaPetPend().add(es(mu,lambda));
    double[] punto=new double[2];
    punto[0]=coef;
    punto[1]=es(mu,lambda);
    this.getListaPuntos().add(punto);
    salida+="*****\n";
}
return salida;
}

/**

```

```

    * @return the listaCoef
    */
    public java.util.ArrayList getListaCoef() {
        return listaCoef;
    }

    /**
     * @return the listaPetPend
     */
    public java.util.ArrayList getListaPetPend() {
        return listaPetPend;
    }

    /**
     * @return the listaPuntos
     */
    public java.util.ArrayList<double[]> getListaPuntos() {
        return listaPuntos;
    }

}

package subsistdiscovariaspet;

import subsistdiscovariaspet.*;

/**
 *
 * @author Lea
 */
public class AnalisisDeVariasPet extends javax.swing.JApplet {

    /** Initializes the applet AnalisisDeVariasPet */
    public void init() {
        try {
            java.awt.EventQueue.invokeAndWait(new Runnable() {
                public void run() {
                    initComponents();
                }
            });
        } catch (InterruptedException ex) {
            java.awt.EventQueue.invokeLater(new Runnable() {
                public void run() {
                    initComponents();
                }
            });
        }
    }
}

```



```

jDialog1Layout.setVerticalGroup(
    jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jDialog1Layout.createSequentialGroup()
            .addGap(26, 26, 26)
            .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 50,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jButton2)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

jTextArea1.setColumns(20);
jTextArea1.setEditable(false);
jTextArea1.setRows(5);
jTextArea1.setWrapStyleWord(true);
jScrollPane1.setViewportView(jTextArea1);

jLabel1.setFont(new java.awt.Font("Tahoma", 1, 14));
jLabel1.setText("Mu");

jTextField1.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
jTextField1.setText("40");

jLabel2.setFont(new java.awt.Font("Tahoma", 1, 14));
jLabel2.setText("Lambda");

jTextField2.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
jTextField2.setText("20");

jTextField3.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
jTextField3.setText("3");

jLabel3.setFont(new java.awt.Font("Tahoma", 1, 14));
jLabel3.setText("Peticiones \nPendientes");

jPanel1.setBackground(new java.awt.Color(255, 255, 255));
jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 462, Short.MAX_VALUE)
    );
jPanel1Layout.setVerticalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 217, Short.MAX_VALUE)
    );

jButton1.setFont(new java.awt.Font("Tahoma", 1, 12));
jButton1.setText("Calcular");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jLabel5.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
jLabel5.setText("Detalles de los calculos realizados");

```



```

        .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(67, 67, 67))
    .addGroup(layout.createSequentialGroup())
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel1)
            .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(28, 28, 28)
        .addComponent(jLabel2)
        .addGap(37, 37, 37)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 33,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 31,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(11, 11, 11)
        .addComponent(jLabel6)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(8, 8, 8)
        .addComponent(jLabel8)
        .addContainerGap())
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup())
        .addContainerGap(368, Short.MAX_VALUE)
        .addComponent(jLabel7)
        .addGap(151, 151, 151))
    );
} // </editor-fold>

```

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    double mu, lambda;
    int i;
    try{
        mu=Integer.parseInt(this(jTextField1.getText()));
        lambda=Integer.parseInt(this(jTextField2.getText()));
        if(mu>lambda){
            i=Integer.parseInt(this(jTextField3.getText());
            Analisis a = new Analisis(mu,lambda,i);
            this.jTextArea1.setText(a.salida());
            this.listaValores=a.getListaPuntos();
            dibujar();
        }else{
            this.jLabel4.setText("Mu debe ser mayor a Lambda");
            this.jDialog1.setSize(300,180);
            this.jDialog1.setLocation(300,200);
            this.jDialog1.setVisible(true);
        }
    } catch(Exception e){
        // javax.swing. // dia=new javax.swing.JDialog();
        this.jLabel4.setText("Error en los parametros");
        this.jDialog1.setSize(300,180);
        this.jDialog1.setLocation(300,200);
    }
}

```

```

        this.jDialog1.setVisible(true);
    }
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.jDialog1.hide();
}

private void dibujar() {

    double XMax=-10000;
    double YMax=-10000;
    for(int i=0;i<this.listaValores.size();i++){
        if(XMax < this.listaValores.get(i)[0])
            XMax=this.listaValores.get(i)[0];
        if(YMax < this.listaValores.get(i)[1])
            YMax=this.listaValores.get(i)[1];
    }
    java.awt.Graphics g = jPanel1.getGraphics();
    java.awt.Dimension d = jPanel1.getSize();
    apAncho = d.width;
    apAlto = d.height;
    g.setColor(java.awt.Color.WHITE);
    g.fillRect(0, 0, apAncho, apAlto);
    g.setColor(java.awt.Color.BLACK);
    g.drawRect(0, 0, apAncho-1, apAlto-1);
    xmin = -0.2;
        xmax = (XMax+0.03);
        ymin = -0.2;
        ymax = (YMax+0.03);
    int apCero=(int)( (0-xmin) * (apAncho-1) / (xmax-xmin) );
    int ypCero= (int)( (0-ymin) * (apAlto-1) / (ymax-ymin) );
    ypCero = apAlto - ypCero;
    g.setColor(java.awt.Color.black);
    g.drawLine(0,ypCero,(int)apAncho,ypCero);
    g.drawLine(apCero,apAlto,apCero,0);
    int xp,yp;
    //yp = (int)( (0.5-ymin) * (apAlto-1) / (ymax-ymin) );
    xp=(int)( ( 0.5 - xmin) * (apAncho-1) / (xmax-xmin) );
    g.drawLine(apCero, ypCero, xp, ypCero);

    for(int i=0;i<this.listaValores.size();i++){
        double XP;
        XP=this.listaValores.get(i)[0];
        xp=(int)( (XP - xmin) * (apAncho-1) / (xmax-xmin) );
        yp=(int)ValorY1(i);
        g.setColor(java.awt.Color.orange);
        g.fillOval(xp, yp, 7, 7);
    }
    double XP=0;
    while(XP < XMax){
        xp=(int)( ( XP - xmin) * (apAncho-1) / (xmax-xmin) );
        g.setColor(java.awt.Color.BLUE);
        String s="" +XP;
        s=s.substring(0,3);
        g.drawString(s,(int)xp,(int)ypCero+10);
    }
}

```

```

        XP +=0.2;
    }
    double YP=0;
    while(YP <YMax){

        g.setColor(java.awt.Color.BLUE);
        String s="" +YP;
        s=s.substring(0,3);
        g.drawString(s,(int)apCero,(int)ValorY2(YP));
        YP +=0.2;
    }
}
private int ValorY1(int valor) {
    double x, y;
    int retorno;
    x = (valor * (xmax - xmin) / (apAncho - 1)) + xmin;
    y=this.listaValores.get(valor)[1];
    retorno = (int) ((y - ymin) * (apAlto - 1) / (ymax - ymin));
    retorno = apAlto - retorno;
    return (retorno);
}
private int ValorY2(double valor) {
    double y;
    int retorno;
    y=valor;
    retorno = (int) ((y - ymin) * (apAlto - 1) / (ymax - ymin));
    retorno = apAlto - retorno;
    return (retorno);
}

// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JDialog jDialog1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JPanel jPanel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTextArea jTextArea1;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField2;
private javax.swing.JTextField jTextField3;
// End of variables declaration
private int apAlto,apAncho;
private double xmax,xmin,ymax,ymin;
private java.util.ArrayList<double[]> listaValores;
}
package subsistdiscovariaspet;

/**
 *
 * @author Lea
 */

```

```
public class Prueba {
    public static void main(String arg[]){
        Analisis a=new Analisis(40,20,3);
        System.out.print(a.salida());
    }
}

package subsistdiscovariaspet;

/**
 *
 * @author Lea
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        /*Analisis a=new Analisis(40,20,3);
        System.out.print(a.salida());*/
        AnalisisDeRend ven=new AnalisisDeRend();
        ven.setVisible(true);
    }
}
```

## Datos y ejecuciones

Al ejecutar el applet, comenzamos con unos datos predefinidos de  $\mu$  ( $\mu$ ),  $\lambda$  ( $\lambda$ ) y las peticiones pendientes, estos datos pueden ser modificados por el usuario siempre que sean validos de acuerdo a la teoría de colas.

También podremos ver los resultados obtenidos en cada paso de la simulación en el panel superior de la ventana principal y un gráfico estadístico en el panel inferior de la misma ventana.

Detalles de los cálculos realizados

Mu

Lambda

Peticiones Pendientes

Calcular

Gráfico Estadístico

Esperas

Coeficientes

Ventana Principal

Panel de detalles de los pasos

Valores para la simulación

**Detalles de los calculos realizados**

Lista de Peticiones Pendientes:  
{ 0.1; 0.30000000000000004; 0.5; 0.7000000000000001; 0.9; 0.05;  
0.55; 0.65; 0.75; 0.8500000000000001; 0.9500000000000001; 0.0  
0.36666666666666664; 0.4333333333333335; 0.5; 0.566666666  
0.275; 0.325; 0.375; 0.4250000000000004; 0.475000000000000

Lista de Pares Ordenados:  
{ (0.1, 0.1); (0.3, 0.30000000000000004); (0.5, 0.5); (0.7, 0.7000000  
(0.55, 0.55); (0.65, 0.65); (0.75, 0.75); (0.85, 0.8500000000000001)  
(0.36666666666666664, 0.36666666666666664); (0.43333333333  
(0.275, 0.275); (0.325, 0.325); (0.375, 0.375); (0.425, 0.425000000

**Mu**

**Lambda**

**Peticiones Pendientes**

**Calcular**

**Gráfico Estadístico**

Esperas

Gráfico estadístico

Botón para el inicio de la simulación

## **Resultados y conclusiones**

Los resultados obtenidos con el applet desarrollado son similares a los del programa desarrollado para el SistOper como era de esperarse, además, se verificó lo visto en teoría en cuanto a las diferencias en tiempos en cola de espera en disco de las distintas peticiones, según las distintas cargas de trabajo simuladas.

La modalidad implementada de mostrar los resultados paso a paso por pantalla permite observar el comportamiento del algoritmo y facilita la comprensión de su funcionamiento.

Como era de esperarse, se observa que en todos los casos, la forma de la curva de los gráficos obtenidos es similar, apreciándose un muy buen ajuste de los valores obtenidos (y graficados) a una recta.

Asimismo se observa un leve impacto en los tiempos de espera ante incrementos de la carga de trabajo, expresada según los coeficientes considerados, obteniéndose que el número promedio de peticiones pendientes tiende a 1.

## **Bibliografía**

D. L. la Red Martinez. **Sistemas Operativos**. EUDENE – UNNE, Argentina, 2004.

## **Página Web:**

<http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/SOF.htm>