SIMULACIÓN DE ALGORITMO DE PLANIFICACIÓN EN ARQUITECTURA MULTIPROCESADOR

Yari Ivan Taft, UTN-FRBA, ivan94@gmail.com
Gabriel Browarnik, UTN-FRBA, gabrielbrowar@gmail.com
Guido Luca Oliveri, UTN-FRBA, guidolucaoliveri@gmail.com
Jose Luis Rodriguez Guia, UTN-FRBA, joserpg94@gmail.com

Resumen— Los servidores suelen tener que calcular cuántos procesadores necesitan incluir para poder satisfacer las necesidades de un cliente. Las necesidades del cliente son las que determinaran el tiempo de atención de los procesos (TA) y el intervalo entre arribos de procesos (IA).

Esta simulación tiene como objetivo determinar qué cantidad de procesadores es la óptima en cuestiones de costos, eficiencia en el uso de los procesadores y atendiendo a las reglas de negocios más habituales.

Se simularon tres posibles escenarios en los cuales en cada uno se cambió la cantidad de procesadores del servidor, y se llegó a la conclusión que la cantidad optima de procesadores es 4. Contemplando la posible futura adquisición de más procesadores de ser necesarios, ya que la arquitectura de servidor Blade [0] lo permite.

Palabras clave— *Arquitectura Multiprocesador, Costo de Procesamiento, Planificación, Servidor Blade*

1. Introducción

Antes de implementar un servidor Blade se hace una simulación con entradas aleatorias de procesos en un algoritmo de planificación para ver los tiempos de respuesta, tiempos ociosos de los procesadores, tiempos de espera promedio, el costo de cada procesador, el costo por segundo de procesamiento y el costo de la energía consumida tanto cuando el procesador está ejecutando un proceso como cuando se encuentra en estado ocioso.

Todos los procesos deben ser atendidos.

Cada procesador tiene su cola asignada, y no se permite el paso de un proceso de una cola a otra puesto que el algoritmo de planificación (FIFO) es no apropiativo.

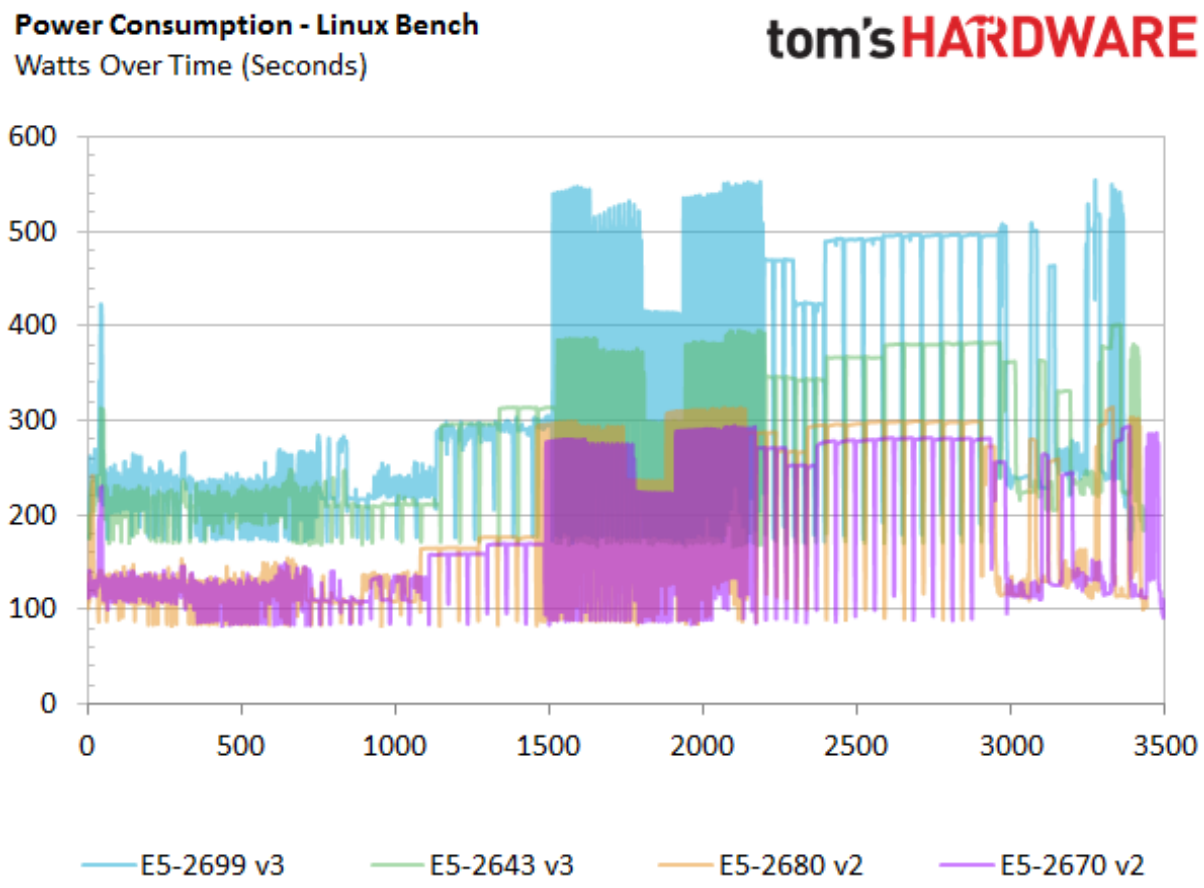
Se conoce el Tiempo de atención cuando el proceso empieza a ser ejecutado, las funciones de densidad de probabilidad son distribuciones Power Function con rango entre 6 y 12 milisegundos para el Intervalo entre Arribos y con rango entre 0,022 y 0,045 segundos para Tiempo de atención [1]. También como regla de negocio se debe respetar que el tiempo de respuesta promedio del sistema sea menor a 1 segundo.

La distribución de procesos se realizará buscando la cola con menor cantidad de procesos en caso de empate se le asigna a la cola de mayor subíndice.

2. Materiales y Métodos

Para realizar la simulación se utilizó un procesador Intel Xeon E5-2670 como procesador de referencia, el costo de cada procesador es de 4900$ [2] y también se calcularon los costos relacionados con los watts utilizados por el procesador, los cuales varían entre 84 watts cuando se encuentra procesando en modo ocioso (IDLE) y 290 watts cuando se encuentra realizando procesamiento útil [3] luego para calcular dicho costo se tuvo en cuenta el precio del kWatt el cual es $4,5 [4].

Tabla 0. Consumo eléctrico de Procesadores Xeon 2600



Fuente: http://www.tomshardware.com

Tabla 1. Clasificación de Variables

Análisis previo	
Metodología	Evento a Evento
Clasificación de variables	
Datos	TA (i) (tiempo de atención en cada puesto), IA (intervalo entre arribos del proceso)
Control	NP (número de procesadores), NC (número de colas)
Resultado	PPU= Porcentaje Total de procesamiento útil del sistema CTSPU=Costo Total del segundo de procesamiento útil del sistema TR2= Tiempo de Respuesta promedio por proceso
Estado	NPS (i) (número de procesos en cada cola+proceso en ejecución)

Fuente: Elaboración Propia

Tabla 2. Tabla de Eventos

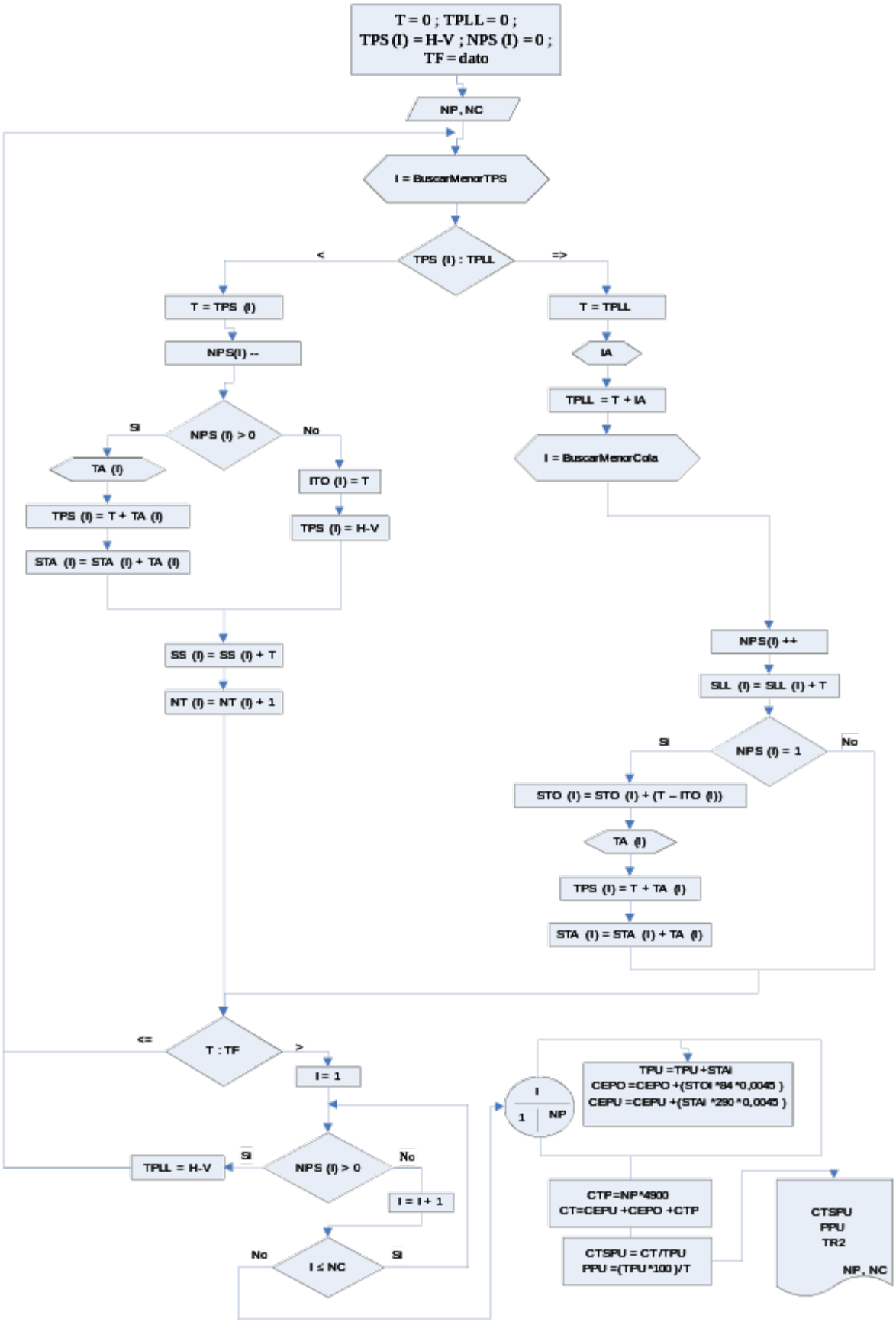
		TEI		
TEF	EVENTO CONDICIÓN	EFNC	EFC	
TPLL	Llegada	Llegada	Salida (i)	NPS (i) = 1
TPS (i)	Salida (i)	-	Salida (i)	NPS (i) > 0

Fuente: Elaboración Propia

2.1 Aclaraciones de Variables Utilizadas para llegar a los resultados:

TPU=Tiempo de procesamiento útil.
CEPO=Costo eléctrico de procesamiento ocioso.
(Watts consumido por segundo ocioso * segundos de procesamiento ocioso*costo del watt)
CEPU=Costo eléctrico de procesamiento útil.
(Watts consumido por segundo útil * segundos de procesamiento útil*costo del watt)
CTP=Costo total de los procesadores.
CT=Costo total (la suma de los 3 costos anteriores)

Figura 1. Diagrama de Flujo



Fuente: Elaboración Propia

2.2 Aclaraciones de Variables Utilizadas para llegar a los resultados:

TPU=Tiempo de procesamiento útil.

CEPO=Costo eléctrico de procesamiento ocioso.

$$(Watts\ consumido\ por\ segundo\ de\ procesamiento\ ocioso * segundos\ de\ procesamiento\ ocioso * costo\ del\ watt) \quad (1)$$

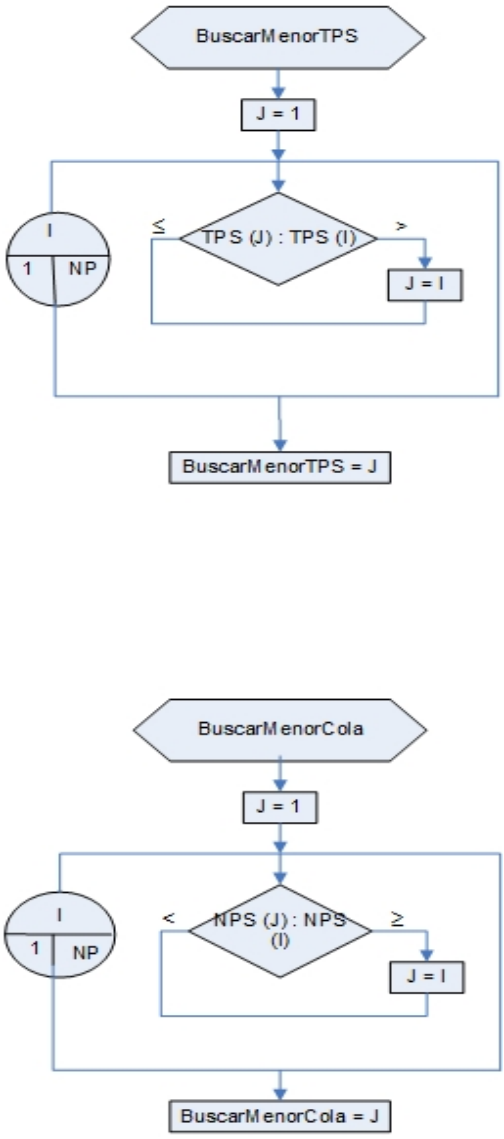
CEPU=Costo eléctrico de procesamiento útil.

$$(Watts\ consumido\ por\ segundo\ de\ procesamiento\ útil * segundos\ de\ procesamiento\ útil * costo\ del\ watt) \quad (2)$$

CTP=Costo total de los procesadores.

CT=Costo total (la suma de los 3 costos anteriores)

Figura 2. Rutinas de BuscarMenosTPS y BuscarMenorCola



Fuente: Elaboración Propia

2.3 Funciones de densidad de Probabilidad

Las funciones de densidad de probabilidad(FDP) fueron obtenidas analizando valores aleatorios con el software llamado EasyFit [5] con una distribución de tipo Power Function [6] y luego se calculó la inversa de la función de distribución acumulativa desde el software WolframAlpha [7].

Función de densidad probabilidad en distribución Power Function:

$$f(x) = \frac{\alpha (x-a)^{\alpha-1}}{(b-a)^{\alpha}} \quad (3)$$

Función de distribución acumulativa en distribución Power Function:

$$F(x) = \left(\frac{x-a}{b-a}\right)^{\alpha} \quad (4)$$

Intervalo entre Arribos

A partir de 500 valores aleatorios entre 6 y 12 milisegundos se obtienen los siguientes parametros:

$\alpha = 0.88277 \; a = 0{,}006 \; b = 0{,}012$

Función de densidad de probabilidad:

$$fdp(x) = \frac{0.88277\,(x-0{,}006)^{0.88277-1}}{(0{,}012-0{,}006)^{0.88277}} \quad (5)$$

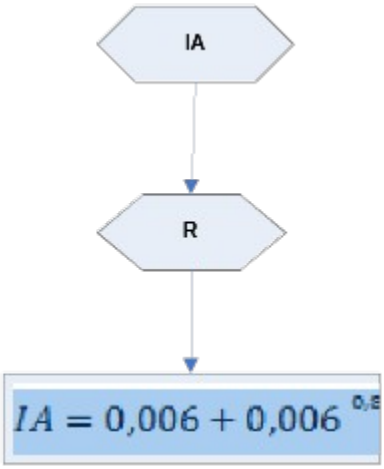
Función acumulada

$$F(X) = \left(\frac{x-0{,}006}{0{,}012-0{,}006}\right)^{0.88277} \quad (6)$$

Por método de la inversa se llega a la ecuación de IA:

$$IA = 0{,}006 + \left(0{,}006 * \sqrt[0.88277]{R}\right) \quad (7)$$

Figura 3. Rutina de Generación de Intervalo entre Arribos(IA)



Fuente: Elaboración Propia

Tiempo de atención

Tomando procesos con una rango de instrucciones entre 600.000.000 y 1.200.000.000 y teniendo en cuenta que nuestro procesador ejecuta 3.300.000.000*8 instrucciones por segundo (3.3 GHz * 8 núcleos) [2] se deduce que el tiempo de atención de cada proceso oscila entre los 0,022 segundos y los 0,045 segundos.

Se generaron 500 valores aleatorios entre 0,022 y 0,045:

$\alpha = 0{,}94342 \; a = 0{,}022 \; b = 0{,}045$

Función de densidad de probabilidad:

$$fdp(x) = \frac{0{,}94342(x - 0{,}022)^{0{,}94342-1}}{(0{,}045 - 0{,}022)^{0{,}94342}} \qquad (8)$$

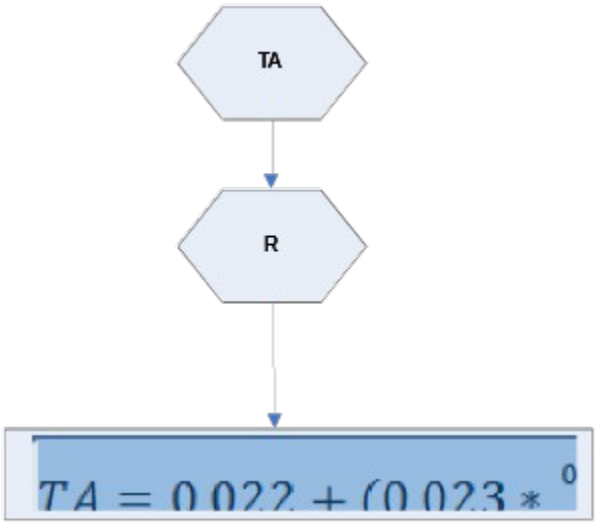
Función acumulada

$$F(X) = \left(\frac{x - 0{,}022}{0{,}045 - 0{,}022}\right)^{0{,}94342} \qquad (9)$$

Por método de la inversa se llega a la ecuación de TA:

$$TA = 0{,}022 + \left(0{,}023 * \sqrt[0{,}94342]{R}\right) \qquad (10)$$

Figura 4. Rutina de Generación de Tiempo de Atención(TA)



Fuente: Elaboración Propia

Modelo Computacional (Java Script)

```
var HV;
var T;
var TF;
var TPLL;
var TPS;
var NPS;
var ITO;
var SS;
var NT;
var STA;
var SLL;
var STO;
var TPU;
var CEPO;
var CEPU;
var CTSPU;
var PPU;
var NP;
var TR;
var NT2;
var TR2;
```

```
function init_variables(tiempo_fin){
    HV = 99999999999999999999;
    T = 0;
    TF = tiempo_fin;
    TPLL = 0;
    TPS = [];
    NPS = [];
    ITO = [];
    SS = [];
    NT = [];
    STA = [];
    SLL = [];
    STO = [];
    TPU = 0;
    CEPO = 0;
    CEPU = 0;
    CTSPU = 0;
    PPU = 0;
    NP = 0;
    TR=0;
    NT2=0;
    TR2=0;
}

function asignar_Tiempos(cTPS){//, cNPS){
  for (var i=0; i<cTPS; i++){
    TPS.push(HV);
    NPS.push(0);
    ITO.push(0);
    SS.push(0);
    NT.push(0);
    STA.push(0);
    SLL.push(0);
    STO.push(0);
  }
  //for (i=0, i<cNPS, i++){
  //  NPS[i]=HV;
  //}
}
```

```
function buscarMenorTPS(){
  var j=0;
  for (i=0;i<NP;i++){
    if(TPS[j]>TPS[i]){
      j=i;
    }
  }
  return j;
}

function buscarMenorCola(){
  var j=0;
  for (i=0;i<NP;i++){
    if(NPS[j]>NPS[i]){
      j=i;
    }
  }
  return j;
}

function getTA() {
  var R = Math.random();
  return (Math.pow(R,(1/0.94342))*0.023)+0.022;
}

function getIA() {
  var R = Math.random();
  return (Math.pow(R, (1/0.88277))*0.006) + 0.006;
}

function hayVaciamiento(){
  var rep = true;
  var i = 0;
  while (rep){
    if(NPS[i]>0){
      TPLL=HV;
      return true;
    } else {
      i++;
```

```
      if(i<NP){
        //cicla otra vez
      } else {
        return false;
      }
    }
  }
}

function calculos(){
  for(var i=0;i<NP;i++){
    TPU += STA[i];
    CEPO += STO[i]*84*0.0045;
    CEPU += STA[i]*290*0.0045;
    NT2=NT2+NT[i];
    TR=SS[i]-SLL[i];
  };
  var CTP = NP * 4900;
  CT = CEPU + CEPO + CTP;
  CTSPU = CT/TPU;
  PPU = (TPU/(T*NP))*100;
  TR2=TR/NT2;
}

function mostrarResultados(){
  console.log("CTSPU = "+ CTSPU);
  console.log("PPU = "+ PPU + "%");
  console.log("Tiempo de Respuesta promedio: "+TR2);
 //console.log("Tiempo de Procesamiento Util: "+TPU);
}

function mostrarEstado(i){
  console.log("T = "+T);
  console.log("TPLL = "+TPLL);
  console.log("TPS["+i+"] = "+TPS[i]);
  console.log("NPS["+i+"] = "+NPS[i]);
  console.log("NT = "+NT);
  console.log("STO = ["+STO + "]");
}
```

```
function calcularSTO(){
  for (i=0;i<NP;i++){
    STO[i] = T - STA[i];
  }
}

function main(cProc, final){
  init_variables(final);
  NP=cProc;
  asignar_Tiempos(NP);
  var repetir = true;
  while (repetir){
    var i = buscarMenorTPS();
    if(TPS[i] < TPLL){
      console.log("-------------");
      console.log("Desencola");
      mostrarEstado(i);
      T = TPS[i];
      calcularSTO();
      NPS[i]--;
      if (NPS[i]>0){
        //STO[i] += (T - ITO[i]);
        STO[i] = (T - STA[i]);
        console.log("-------------");
        console.log("Atiende");
        mostrarEstado(i);
        TA = getTA();
        TPS[i] = T +TA;
        STA[i] += TA;
      } else {
        console.log("Inicio tiempo ocioso");
        ITO[i] = T;
        TPS[i] = HV;
      }
      SS[i] += T;
      NT[i] += 1;
    } else {
      console.log("-------------");
      console.log("Llega tarea");
```

```
      mostrarEstado(i);
      T = TPLL;
      IA = getIA();
      TPLL = T +IA;
      i = buscarMenorCola();
      NPS[i]++;
      SLL[i] += T;
      if (NPS[i]==1){
        //STO[i] += (T - ITO[i]);
        STO[i] = (T - STA[i]);
        console.log("-------------");
        console.log("Atiende");
        mostrarEstado(i);
        TA = getTA();
        TPS[i] = T + TA;
        STA[i] += TA;
      } else {
        console.log("Hace cola");
      }
    }
    console.log("-------------");
    console.log("Fin ciclo");
    mostrarEstado(i);
    if (T >=TF){
      if (hayVaciamiento()){
        repetir=true;
      } else {
        repetir = false;
      }
    }
  }
  calcularSTO();
  calculos();
  mostrarResultados();
}
```

3. Resultados y Discusión

3.1 Resultados

Se ejecutó la simulación en 3 escenarios posibles:

Sistema de 1 procesadores ejecutando procesos por 25 segundos

CTSPU = 53.018813788781195

PPU = 100%

Tiempo de Respuesta promedio: 34.95191428094066 segundos

Sistema de 2 procesadores ejecutando procesos por 25 segundos

CTSPU = 105.7231995997371

PPU = 99.80324607547425%

Tiempo de Respuesta promedio: 5.484596826989113 segundos

Sistema de 4 procesadores ejecutando procesos por 25 segundos

CTSPU = 210.69084120583852

PPU = 93.47297002011015%

Tiempo de Respuesta promedio: 0.007845222464231468 segundos

3.2 Discusión:

De los resultados se puede asumir que la mejor alternativa para la frecuencia y duración de los procesos es un servidor de 4 microprocesadores ya que con 1 y 2 procesadores no se cumple la regla de negocio de un tiempo de respuesta menor 1 segundo.

Si en un caso hipotético se hubiesen utilizado más procesadores con las mismas condiciones, podríamos ver rápidamente que el costo del segundo de procesamiento útil se incrementaría pero habría a la vez procesadores con una cantidad notable de tiempo ocioso, lo cual resultaría en un desperdicio tanto de energía como de gasto en hardware.

Si se proyecta que a futuro se quieren introducir procesos con mayor frecuencia podría parecer que la opción más acertada es la de sumar nuevos procesadores, ya que los mismos de tipo Blade permiten agregar procesadores a medida que el cliente lo vaya necesitando por lo tanto se pueden comprar 4 y conforme se vaya necesitando más capacidad de procesamiento se pueden ir agregando 1 a 1.

4. Conclusiones y recomendaciones

4.1 Conclusión

Para las necesidades del sistema actual la mejor alternativa es comprar 4 procesadores, e ir agregando conforme se vaya necesitando.

Es lo más económico en tanto costo por segundo de procesamiento útil, lo más eficiente en tanto porcentaje de tiempo no ocioso y cumple con la regla de negocio.

4.2 Recomendaciones:

1) A la hora de realizar una simulación de este estilo es muy importante tener en cuenta cómo se van a calcular los tiempos de atención y los intervalos entre arribos ya que esta información debe ser obtenida mediante un estudio específico de las necesidades del cliente.
2) Los servidores son escalables, esto significa que si un servidor no tiene la suficiente capacidad de procesamiento no hay que comprar uno nuevo, sino que únicamente hay que comprarle más procesadores para agregárselos y así obtener mayor capacidad de procesamiento.
3) La simulación aborda un microprocesador en particular, de querer implementar la simulación con otro microprocesador se deberá hacer un estudio del consumo eléctrico y un estudio de mercado para conocer el costo del mismo.
4) Se debe actualizar el valor del watt al día de la fecha si se quiere implementar esta simulación.
5) Puede parecer que ser que una simulación de 25 segundos es bastante corta pero en realidad en tiempos computacionales en 25 segundos pueden realizar millones de operaciones por lo que no es necesario ejecutar una simulación con un tiempo muy largo.
6) Esta simulación solo maneja un algoritmo de planificación first in first out (FIFO), si el sistema operativo sobre el cual se trabajará maneja otro tipo de algoritmo hay que modificar el encolado de los procesos en el algoritmo.

5. Referencias

[0] https://en.wikipedia.org/wiki/Blade_server

[1] P. MARTINEZ, M. CABELLO, J.C. DIAZ MARTIN (1997). Sistemas Operativos. Editorial: Diaz de Santos S.A. Página 71-73.

[2] Costo del Procesador Intel® Xeon® E5-2670:

http://articulo.mercadolibre.com.ar/MLA-625145409-intel-xeon-i7-2011-e5-2670-8-cores16-threads-20mb-cache-_JM

[3] Consumo eléctrico en IDLE y en procesamiento útil del Intel Xeon E5-2670:

http://www.tomshardware.com/reviews/intel-xeon-e5-2600-v4-broadwell-ep,4514-8.html

[4] Costo del kWatt en Argentina 2017:

http://www.edesur.com.ar/cuadro_tarifario.pdf Pagina 2 Tarifa 3, Cargo por Potencia Adquirida.

[5] Distribución Power Function:
Mirza Naveed-Shahzad, Zahid Asghar, Farrukh Shehzad, Mubeen Shahzadi (2015)
Parameter Estimation of Power Function Distribution with TL-moments. Página 321.

[6] http://www.mathwave.com/es/home.html

[7] https://www.wolframalpha.com/