INSTITUTO POLITECNICO NACIONAL

CENTRO DE INVESTIGACION EN COMPUTACION

"DESARROLLO DE UN SISTEMA GRID COMPUTING PARA LA DISTRIBUCION DE SERVICIOS EN LABORATORIOS VIRTUALES"

TESIS

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS DE LA COMPUTACION

PRESENTA:

ING. JOSUE RANGEL GONZALEZ

DIRECTOR DE TESIS: DR. LUIS ALFONSO VILLA VARGAS
DIRECTOR DE TESIS: DR. JOSE LUIS OROPEZA RODRIGUEZ

MÉXICO D.F. JULIO DEL 2009

INSTITUTO POLITECNICO NACIONAL

SECRETARIA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 14:00 horas del día 4 del mes de Junio de 2009 se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis de grado titulada:

"DESARROLLO DE UN SISTEMA GRID COMPUTING PARA LA DISTRIBUCIÓN DE SERVICIOS EN LABORATORIOS VIRTUALES"

RANGEL	**GÓNZALEZ**	**JOSUE**
Apellido paterno	materno	nombre(s)

Con registro:

A	0	7	0	2	4	3

aspirante al grado de: ***MAESTRÍA CIENCIAS DE LA COMPUTACIÓN***

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***SU APROBACIÓN DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Presidente

Dr. Ricardo Barrón Fernández

Secretario

M. en C. Alejandro Botello Castillo

Primer vocal
(Director de tesis)

Dr. Luis Alfonso Villa Vargas

Segundo vocal
(Director de tesis)

Dr. José Luis Oropeza Rodríguez

Tercer Vocal

M. en C. Felipe Dávalos Rodríguez

Suplente

M. en C. Sandra Dinora Orantes Jiménez

El presidente del Colegio

Dr. Jaime Álvarez Gallegos

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION EN COMPUTACION
DIRECCION

INSTITUTO POLITECNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESION DE DERECHOS

En la Ciudad de México D.F. el día 12 del mes Junio del año 2009, el que suscribe Josué Rangel González alumno del Programa de Maestría en Ciencias de la Computación con número de registro A070243, adscrito al Centro de Investigación en Computación, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del Dr. Luis Alfonso Villa Vargas y del Dr. José Luis Oropeza Rodríguez y cede los derechos del trabajo intitulado Desarrollo de un Sistema Grid Computing para la Distribución de Servicios en Laboratorios Virtuales, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección joshuargmx@hotmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Josué Rangel González

Nombre y firma

RESUMEN

Debido a la gran popularidad de los ambientes de trabajo en red, los emergentes procesadores multi-core (múltiples núcleos) paralelos y de la computación distribuida; se está volviendo necesario el desarrollo de aplicaciones en todos los niveles. Además de ambientes de computación para las computadoras personales en escala-Internet como lo son las Grid (Mallas) y P2P (*peer to peer, punto a punto*).

Aplicaciones como los laboratorios virtuales pueden ofrecer una gran ayuda a todas aquellas personas que carecen de infraestructura adecuada y recursos económicos para poder comprar equipos costosos con los cuales se pueden ejecutar pruebas hacia un producto que van a lanzar al mercado. Desgraciadamente estos laboratorios no siempre cuentan con una cantidad suficiente de recursos para que una gama de múltiples usuarios puedan hacer uso de sus servicios al mismo tiempo, de esta manera se presentan problemas, ya que la demanda de recursos puede llegar a ser excesiva para un solo servidor, aunque éste cuente con multiprocesamiento y/o procesamiento paralelo. Por tanto, se da paso a una nueva forma de atacar tales problemas mediante la tecnología que ofrece la Grid Computing (Computación en Mallas), está une múltiples computadoras heterogéneas para ofrecer una mayor cantidad de recursos; además de ello, brinda algoritmos de distribución de carga y balanceo de trabajo para ofrecer una mejor capacidad de cómputo.

En base a la problemática planteada surge esta investigación, que propone un sistema Grid Computing basado en .NET para la distribución de servicios en laboratorios virtuales enfocándose específicamente a la ejecución de aplicaciones que son lo bastante intensivas como para realizar esta tarea en una sola computadora aislada. Este tipo de aplicaciones requiere un soporte avanzado para múltiples tareas con relaciones entre las dependencias de datos, además de múltiples módulos, los cuales reciben entradas de datos, realizan operaciones y generan resultados. Por tanto, se plantea ofrecer una solución para hacer frente a este tipo de problemas a lo largo de este trabajo.

Contribuciones como la reducción de tiempo de espera de los usuarios al realizar ejecución de aplicaciones complejas y la utilización de equipos de escritorio heterogéneos para la generación de una Grid, se ofrecen en este proyecto de investigación.

ABSTRACT

Due to the popularity of the work environment on a network, the emerging multi-core processors (multiple cores) parallel and distributed computing is becoming necessary to develop applications at all levels, besides computing environments for personnel computers in Internet-scale such as the Grid and P2P (peer to peer).

Applications such as virtual laboratories can provide a great help to all those who lack adequate infrastructure and financial resources to purchase expensive equipment which can be run to test a product that will launch to market. Unfortunately these laboratories do not always have enough resources for a range of multiple users to use their services at the same time; this would present problems, since the demand for resources may be excessive for a single server. Although this has multiprocessing and / or parallel processing, thus giving way to a new way to tackle such problems through technology offered by Grid Computing, is joining multiple heterogeneous computers to offer more resources, it also provides algorithms for load distribution and balancing provide improved computing.

Based on this research appears problematic, which proposes a system based on Grid Computing. NET for the distribution of services in virtual laboratories focusing specifically on the implementation of applications that are sufficiently intensive to perform this task on one computer alone, this application requires an advanced support for multiple tasks with relationships between units of data including multiple modules, which receive data inputs, operations and generate results. Therefore there is a solution to address these problems through this work.

Contributions such as reducing time for users to make and implement complex applications using heterogeneous desktops for the generation of a Grid are offered through this research project.

AGRADECIMIENTOS

A mis padres. Rogelio y Carmen

Por su apoyo incondicional en todo momento, por sus concejos y regaños ya que con cada uno de ellos me hicieron madurar y me ayudaron a darme cuenta que la vida hay que vivirla paso a paso. Gracias por su confianza y Amor

A mi hermano. Adrian

Por creer en mí y por ayudarme en los momentos difíciles de mi vida, gracias por demostrarme que siempre tendré tu apoyo.

A mi esposa. Nancy

Por brindarme los momentos más hermosos de mi vida, por tu fuerza, alegría y cariño, el cual siempre te encargas de transmitirme, por crear en mi una sonrisa hasta en los momentos más difíciles, por tu apoyo y por dejarme compartir toda mi vida a tu lado, sobre todo por el Amor que me brindas día con día.

A mi hija. Melanie

Ya que con tu llegada a nuestras vidas nos haz brindado la alegría más grande del mundo, porque desde ese momento eres para nosotros el motivo más importante para seguir adelante persiguiendo nuevas metas.

A mi primo. Ismael

Por tus consejos, ayuda académica y sobre todo por ser un buen amigo que nunca me diste la espalda cuando necesitaba de tu ayuda, por no odiarme ya que siempre te ganaba cuando jugábamos tenis de mesa.

A mis familiares y amigos

Por tenerme paciencia en todo momento, ya que en muchas ocasiones deje de brindarles tiempo en momentos de convivencia, por creer en mí y por compartir todos aquellos momentos significativos en mi vida.

A mis asesores. Dr. Luis Alfonso Villa Vargas y José Luis Oropeza Rodríguez

Por todo el tiempo que me brindaron ayudándome en el desarrollo de este proyecto de investigación, por sus consejos y ayuda.

Al personal de la UTE

En especial a Elda y a Silvia, porque siempre me atendieron con una sonrisa y me trataron como a un amigo, muchas gracias por sus atenciones.

Al CONACYT y a la SIP (Secretaría de Investigación y Posgrado) por el apoyo brindado para la realización de mis estudios. Al Instituto Politécnico Nacional por brindarme la oportunidad de realizar un postgrado de calidad. Gracias al Centro de Investigación en Computación por darme un mundo de conocimiento y por ser mi hogar durante más de 2 años.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

GLOSARIO

Balance o balanceo de carga: Se refiere a la técnica usada para compartir el trabajo a realizar entre varios procesos, ordenadores, discos u otros recursos.

Circuitos VLSI: Integración en escala muy grande de sistemas de circuitos basados en transistores.

Computación Distribuida: Es un nuevo modelo para resolver problemas de computación masiva utilizando un gran número de computadoras organizadas en racimos incrustados en una infraestructura de telecomunicaciones distribuida.

Escalabilidad: En telecomunicaciones y en ingeniería informática, es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.

Middleware: Es un software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas. Funciona como una capa de abstracción de software distribuida, que se sitúa entre las capas de aplicaciones y las capas inferiores (sistema operativo y red).

Modelado UML: El Lenguaje de Modelado Unificado (UML:Unified Modeling Language), es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos que aparecen a fines de los 80's y principios de los 90s.UML es un lenguaje de modelado.

Objeto: Se define como la unidad que en tiempo de ejecución realiza las tareas de un programa.

Proceso: Es un concepto manejado por el sistema operativo que consiste en el conjunto formado por:

- Las instrucciones de un programa destinadas a ser ejecutadas por el microprocesador.
- Su estado de ejecución en un momento dado, esto es, los valores de los registros de la CPU para dicho programa.
- Su memoria de trabajo, es decir, la memoria que ha reservado y sus contenidos.
- Otra información que permite al sistema operativo su planificación.

Programación Orientada a Objetos: Es un paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo y encapsulamiento.

Protocolos: Es un método estándar que permite la comunicación entre procesos (que potencialmente se ejecutan en diferentes equipos), es decir, es un conjunto de reglas y procedimientos que deben respetarse para el envío y la recepción de datos a través de una red.

Proxy: Hace referencia a un programa o dispositivo que realiza una acción en representación de otro. Su finalidad más habitual es la de **servidor proxy**, que sirve para permitir el acceso a Internet a todos los equipos de una organización cuando sólo se puede disponer de un único equipo conectado, esto es, una única dirección IP.

Redes peer to peer (punto a punto): Son redes que aprovechan, administran y optimizan el uso de banda ancha que acumulan de los demás usuarios en una red por medio de la conectividad entre los mismos usuarios participantes de la red, obteniendo como resultado mucho más rendimiento en las conexiones y transferencias con algunos métodos centralizados convencionales.

Simulación: La simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender el comportamiento del sistema.

Script: Es un guión o conjunto de instrucciones. Permiten la automatización de tareas creando pequeñas utilidades. Es muy utilizado para la administración de sistemas UNIX. Son ejecutados por un intérprete de línea de órdenes y usualmente son archivos de texto. También un script puede considerarse una alteración o acción a una determinada plataforma.

DOS: Es una familia de sistemas operativos para PC. El nombre son las siglas de **D**isk **O**perating **S**ystem (sistema operativo de disco). Fue creado originalmente para computadoras de la familia IBM PC, que utilizaban los procesadores Intel 8086 y 8088, de 16 bits y 8 bits, respectivamente, siendo el primer sistema operativo popular para esta plataforma. Contaba con una interfaz de línea de comandos en modo texto ó alfanumérico, por medio de su propio intérprete de órdenes, **command.com.**

La consola DOS: Es la consola que interpreta comandos de DOS en sistemas operativos de Windows. Para acceder se puede buscar dentro del menú de inicio, su ubicación varía según la versión de Windows y las preferencias del usuario.

GLOSARIO

Balance o balanceo de carga: Se refiere a la técnica usada para compartir el trabajo a realizar entre varios procesos, ordenadores, discos u otros recursos.

Circuitos VLSI: Integración en escala muy grande de sistemas de circuitos basados en transistores.

Computación Distribuida: Es un nuevo modelo para resolver problemas de computación masiva utilizando un gran número de computadoras organizadas en racimos incrustados en una infraestructura de telecomunicaciones distribuida.

Escalabilidad: En telecomunicaciones y en ingeniería informática, es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.

Middleware: Es un software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas. Funciona como una capa de abstracción de software distribuida, que se sitúa entre las capas de aplicaciones y las capas inferiores (sistema operativo y red).

Modelado UML: El Lenguaje de Modelado Unificado (UML:Unified Modeling Language), es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos que aparecen a fines de los 80's y principios de los 90s.UML es un lenguaje de modelado.

Objeto: Se define como la unidad que en tiempo de ejecución realiza las tareas de un programa.

Proceso: Es un concepto manejado por el sistema operativo que consiste en el conjunto formado por:

- Las instrucciones de un programa destinadas a ser ejecutadas por el microprocesador.
- Su estado de ejecución en un momento dado, esto es, los valores de los registros de la CPU para dicho programa.
- Su memoria de trabajo, es decir, la memoria que ha reservado y sus contenidos.
- Otra información que permite al sistema operativo su planificación.

Programación Orientada a Objetos: Es un paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo y encapsulamiento.

Protocolos: Es un método estándar que permite la comunicación entre procesos (que potencialmente se ejecutan en diferentes equipos), es decir, es un conjunto de reglas y procedimientos que deben respetarse para el envío y la recepción de datos a través de una red.

Proxy: Hace referencia a un programa o dispositivo que realiza una acción en representación de otro. Su finalidad más habitual es la de **servidor proxy**, que sirve para permitir el acceso a Internet a todos los equipos de una organización cuando sólo se puede disponer de un único equipo conectado, esto es, una única dirección IP.

Redes peer to peer (punto a punto): Son redes que aprovechan, administran y optimizan el uso de banda ancha que acumulan de los demás usuarios en una red por medio de la conectividad entre los mismos usuarios participantes de la red, obteniendo como resultado mucho más rendimiento en las conexiones y transferencias con algunos métodos centralizados convencionales.

Simulación: La simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender el comportamiento del sistema.

Script: Es un guión o conjunto de instrucciones. Permiten la automatización de tareas creando pequeñas utilidades. Es muy utilizado para la administración de sistemas UNIX. Son ejecutados por un intérprete de línea de órdenes y usualmente son archivos de texto. También un script puede considerarse una alteración o acción a una determinada plataforma.

DOS: Es una familia de sistemas operativos para PC. El nombre son las siglas de **D**isk **O**perating **S**ystem (sistema operativo de disco). Fue creado originalmente para computadoras de la familia IBM PC, que utilizaban los procesadores Intel 8086 y 8088, de 16 bits y 8 bits, respectivamente, siendo el primer sistema operativo popular para esta plataforma. Contaba con una interfaz de línea de comandos en modo texto ó alfanumérico, por medio de su propio intérprete de órdenes, **command.com.**

La consola DOS: Es la consola que interpreta comandos de DOS en sistemas operativos de Windows. Para acceder se puede buscar dentro del menú de inicio, su ubicación varía según la versión de Windows y las preferencias del usuario.

Capítulo 1

Introducción

En este capítulo se presenta la descripción del proyecto de investigación, donde se muestra el problema que se pretende resolver; así como los objetivos, solución y metodología propuesta para enfrentar dicho problema.

A continuación se describe el contenido de este capítulo:

- Se describe a la educación virtual y los laboratorios virtuales.
- Posteriormente, se describe el planteamiento del problema.
- Se presentan tanto la justificación como los objetivos a alcanzar.
- Se menciona el alcance del proyecto y las limitaciones que éste presenta.
- Por último, se presenta la estructura que conforma la tesis.

1.1. PLATAFORMAS VIRTUALES

Son escenarios educativos diseñados de acuerdo a una metodología de acompañamiento a distancia. También se pueden ver como herramientas basadas en páginas Web para la organización e implementación de cursos, talleres, laboratorios o para apoyar actividades educativas presenciales.

Este tipo de plataformas se están volviendo un instrumento en forma cada vez más utilizado en ámbitos de enseñanza. Este hecho no solo está relacionado con factores puramente tecnológicos sino también se debe a que gracias a estas plataformas va en aumento la oferta de programas informáticos gracias al vertiginoso desarrollo de la Web; aunado a esto, se deriva la urgente necesidad que tienen los centros académicos de ampliar su oferta educativa, adaptarla a las necesidades de la sociedad y hacerla accesible a un mayor número de estudiantes.

1.2. LA EDUCACIÓN VIRTUAL: ¿UNA ALTERNATIVA A LA EDUCACIÓN TRADICIONAL?

La posibilidad de transmitir conocimientos, imágenes, textos, sonido, recomendaciones, por Internet al hogar, y a una multitud de usuarios simultáneamente a un costo más o menos reducido, y en el futuro a un costo relativamente mínimo, está revolucionando la enseñanza tradicional en sus dos modalidades: presencial y a distancia, y en medio de éstas dos surge una tercera y nueva modalidad de enseñanza: la enseñanza virtual o enseñanza online. Se trata de una forma de enseñanza basada en las nuevas tecnologías de la información y la comunicación fundamentalmente en Internet que promete "revolucionar" la educación tradicional.

En este sentido es interesante analizar si la educación virtual puede sustituir las modalidades tradicionales y por lo tanto saber si presenta o no nuevas y concretas posibilidades de elección para el educando.

Ésta es precisamente la lógica subyacente al desarrollo de propuestas educativas en Internet: las nuevas tecnologías presentan a priori una posibilidad de elección entre la educación presencial y la educación virtual.

En este sentido, a la pregunta: "una nueva opción: ¿para quién?", la educación a distancia a través de Internet responde:

1. Educación virtual para personas sin acceso al sistema educativo tradicional (por ejemplo, personas aisladas geográficamente)
2. Educación virtual para personas con acceso al sistema educativo tradicional.

Ahora bien, una vez definidos los potenciales grupos destinatarios de la educación virtual, deberíamos preguntarnos si la educación virtual es al menos tan efectiva como la educación tradicional.

Claro que esta pregunta no es tan significativa en el primer grupo de destinatarios (personas sin acceso al sistema educativo tradicional) dado que no se trata de una nueva opción (entendiendo *opción* como *alternativa*), sino de una oportunidad sin precedentes.

Gracias a Internet, los estudiantes no tienen que ir físicamente a la escuela, y pueden tener su propio ritmo de estudio de acuerdo a los horarios que más les convenga (aprendizaje asíncrono) lo cual reduce el costo de oportunidad de la educación y permite insertar dentro del mercado educativo a alumnos que nunca antes hubieran podido formarse dentro del sistema educativo tradicional.

De manera que mientras la educación virtual arroje algún resultado positivo (superior a cero) en los indicadores de aprendizaje, debería ser un gran logro, además de que se tienen que recibir con alegría las nuevas oportunidades que las tecnologías de la comunicación y de la información ofrecen para aquellas personas que anteriormente no tenían acceso a ninguna propuesta educativa.

Ahora bien, ¿qué sucede con el segundo grupo de análisis? (personas con acceso al sistema educativo tradicional).

Aquí el análisis de efectividad se vuelve más interesante dado que para promover la elección entre programas de educación virtual y los programas tradicionales debería poder asegurar a ciencia cierta que los primeros son al menos tan efectivos como los segundos.

Y la realidad es que a la pregunta: ¿Qué tan efectiva es la educación vía Internet?

La respuesta es: "TODAVÍA NO SE SABE".

En los últimos años se observa en el mundo una tendencia importante hacia la educación con fines de lucro (la privatización de la educación) y un aumento considerable de la demanda de educación a

distancia. Austan Goolsbee de la Universidad de Chicago, analiza estas dos tendencias y demuestra que si bien hubo un crecimiento rápido de la Internet educativa, su impacto en la productividad educativa no será demasiado grande en los próximos años [1].

Lo que nos ha llevado a pensar… ¿Cómo podemos apoyar de una manera más significativa a todas aquellas personas que tienen o no acceso a la educación tradicional?, en concreto a personas que les interese tener una herramienta que les permita realizar una mayor cantidad de interacciones (por ejemplo: pruebas o simulaciones a cualquier diseño que estén desarrollando en laboratorios de carreras que formen parte del área de las ciencias, ingeniería y/o el comercio), ya que de esta manera obtendrán una percepción de que su interacción con un sistema informático es equivalente al que obtendrían en un esquema presencial tradicional.

1.3. LABORATORIOS VIRTUALES

Se puede definir como un sistema informático que permite operar dentro o fuera de línea, modelar interactivamente lugares, objetos o situaciones con fines de experimentación, investigación y/o observación, contribuyendo a incrementar la capacidad de metodología de búsqueda por el participante, así como de diseño experimental.

Trasladando este entorno a la enseñanza actual, los elementos necesarios para abordar la realización de actividades prácticas son los laboratorios virtuales (LV) y remotos (LR), accesibles a través de Intranet, Internet o ambientes computacionales, donde los usuarios realizan las prácticas de una forma lo más similar posible a como si estuviesen en las instalaciones del laboratorio tradicional (LT), simulando e interactuando con instrumentos virtuales.

En el laboratorio tradicional (LT), los recursos en personas y espacios son restringidos, debido a su masificación y a problemas presupuestarios; se requiere la presencia física del usuario y de un supervisor. Una solución a estos problemas la encontramos, en concreto, en el uso de laboratorios virtuales (LV) y remotos (LR). El LV acerca y facilita la realización de experiencias a un mayor número de alumnos, aunque alumno y laboratorio no coincidan en el espacio. Permite simular fenómenos y modelos físicos, conceptos abstractos, mundos hipotéticos, controlar la escala de tiempo, etc., ocultando el modelo matemático y mostrando el fenómeno simulado de forma interactiva. La creciente complejidad de las actividades en el LT y el desarrollo de las Tecnologías

de Comunicación e Información y la Computación, han hecho que los LV evolucionen, transformándose en laboratorios remotos (LR), donde el usuario utiliza y controla los recursos del laboratorio, a través de una red local (Intranet) o bien a través de Internet [2].

1.4. Áreas de Uso de los Laboratorios Virtuales

Algunas áreas que pertenecen a las ciencias, ingeniería y al comercio están adoptando como un recurso informático justificado a los laboratorios virtuales, por un razonamiento primordial: el reemplazar equipamiento real costoso, por simulaciones de equipamiento conducidas y compartidas a distancia en modalidad sincrónica o asincrónica, en el medio virtual por la gran cantidad de datos que necesidad analizar y procesar, por el alto grado de interacción que ofrecen, por brindar una mayor capacidad de modelación (gráficas tridimensionales).

Además de lo anterior, en los laboratorios virtuales se está volviendo cada día más necesario poder tener en ellos una forma de resolver tareas intensivas, las cuales demandan una gran cantidad de recursos computacionales, como podrían ser, el cálculo de ecuaciones matemáticas complejas, el análisis de fenómenos físicos, etc. Además de ello, existen tanto empresas como estudiantes de ingeniería que crean diseños de productos que posteriormente pueden vender, por lo cual es indispensable para ellos estar completamente seguros de que sus diseños han sido completamente probados en forma exhaustiva antes de lanzarlos al mercado.

1.5. Alternativas para el Incremento de Rendimiento en los Laboratorios Virtuales

- Modelo de paso de mensajes
- Multi-hilos
- Cómputo distribuido
- Grid Computing
- Cualquier combinación de la anteriores

De las cuales se ha seleccionado al Grid Computing (Computación en Malla), ya que esta tecnología se enmarca dentro de la tecnología de computación distribuida. Su funcionalidad principal es la de compartir potencia computacional. Se basa en el aprovechamiento de los

ciclos de procesamiento no utilizados por los millones de ordenadores conectados a la Red. De esta forma se consigue que puedan resolver de forma distribuida tareas que son demasiado intensivas para ser resueltas por una máquina aislada.

1.6. Planteamiento del Problema

Para cubrir la necesidad creciente de demandas en los laboratorios virtuales en cuanto a poder hacer frente a las tareas que son muy demandantes computacionalmente para ser resueltas por una sola computadora aislada, es necesario hacer uso de sistemas de computadoras distribuidos y paralelos [3].

Se plantea para este proyecto de investigación, crear un sistema capaz de realizar cómputo distribuido para disminuir el tiempo de procesamiento de las aplicaciones que requieren resolver tareas intensivas; teniéndose como un caso particular, simplificar la tarea a todas aquellas personas que necesiten realizar pruebas a aplicaciones demandantes en cuanto a cómputo; ya que ellos requieren de un sistema que les permita una ejecución en forma distribuida, para poder generar una mayor cantidad de tareas en menor tiempo.

1.7. Justificación

Las Grid (Mallas) computacionalmente acoplan recursos geográficamente distribuidos, por tanto se está volviendo una computación efectiva para resolver problemas de gran escala en la ciencia, ingeniería y en el comercio [4]. Dado que la computación y la tecnología de comunicación han tenido un impacto significativo en los sistemas de educación. Esta tecnología ha mejorado el aprendizaje colaborativo en línea, además de eso, mejora las experiencias de aprendizaje de los usuarios [5].

Crear un sistema que les permita a los laboratorios virtuales la ejecución de aplicaciones en forma distribuida ayudará a ofrecer a los estudiantes de ingeniería y a las empresas que se dedican a vender sus productos, una herramienta con la cual podrán realizar una cantidad mayor de tareas en tiempos considerablemente menores a los que ahora tienen, logrando con ello lanzar al mercado sus productos más rápidamente, lo que les podría permitir generar mayores ganancias. También puede

lograr reducir los problemas de agotamiento de recursos que se llegan a presentar a la hora de procesar aplicaciones que requieren alto poder computacional.

Día con día está surgiendo la necesidad de trabajar sobre ambientes de diseño electrónico basados en red, por lo cual es necesario que estos ambientes tengan la capacidad de ser escalables, adaptativos, seguros, con alta disponibilidad y bajos costos [5], para que de esta manera se puedan ofrecer mejores servicios.

Por tanto es importante que estos laboratorios puedan ejecutar aplicaciones en forma distribuida, ya que el desarrollo de aplicaciones modernas se está volviendo progresivamente complicado planteando un reto interminable para la ejecución secuencial. Para acomodar la necesidad de disminución del tiempo, está siendo bastante necesario el uso de computación distribuida [6].

Es en este punto la importancia que tiene desarrollar sistemas de laboratorios virtuales que aprovechen la tecnología Computación en Malla, ya que ésta trabaja bajo el esquema de reunir sistemas computacionales heterogéneos no importando su ubicación geográfica bajo una sola malla, para que de esa manera se puedan compartir recursos computacionales y así poder ofrecer reducción en los tiempos de procesamiento.

1.8. OBJETIVOS

1.8.1. OBJETIVO GENERAL

Desarrollar un sistema que les permita a los laboratorios virtuales la ejecución distribuida de procesos en una Grid integrada por computadoras de escritorio (Desktop Computing) heterogénea, utilizando peer to peer (punto a punto) como protocolo de comunicación en el paso de mensajes entre procesos.

1.8.2. OBJETIVOS PARTICULARES

- Desarrollar una aplicación que cree/genere un ambiente estático de una cierta aplicación a ejecutar en el contexto distribuido.

- Desarrollar un modelo para el balance de carga en las unidades de ejecución, tomando en cuenta los requerimientos de cada proceso y recursos disponibles en cada unidad de ejecución.
- Desarrollar un sistema para la distribución de procesos utilizando el modelo de balance de carga.
- Desarrollar un modelo para el paso de mensajes entre procesos utilizando la tecnología punto a punto.
- Desarrollar un sistema de colas para el paso de mensajes entre procesos.
- Implementar un sistema que ocupe el modelo de pasos de mensajes y que inicie la ejecución de los procesos en las unidades de ejecución.

1.9. ALCANCES

El diseño del sistema permitirá ejecutar procesos que conformen una aplicación, reduciendo el tiempo de procesamiento gracias a que se realizará en forma distribuida sobre una malla de computadoras heterogéneas, creada por medio de la tecnología .NET Remoting. Además de ello, tendrá la capacidad de ofrecer el soporte de comunicación punto a punto para múltiples procesos que tienen dependencias de datos. Los usuarios no necesitan preocuparse acerca de los detalles de los procesos, hilos y las comunicaciones explícitas.

La distribución, planificación y balanceo de las cargas de trabajo se realizarán de manera automática y sin la intervención o supervisión de un administrador de sistemas. La presentación de resultados y la generación de la bitácora serán presentados por el sistema al usuario. Además de ello, el sistema en global tendrá la gran ventaja de que puede trabajar con computadoras heterogéneas que formen parte de la Malla.

1.10. LIMITACIONES

No solo la planificación, distribución y balanceo de las cargas de trabajo conforman una solución integral para la reducción en los tiempos de cómputo y en la prevención de la extenuación de recursos; ya que primero se tiene que tener en cuenta que una de las principales bases de este sistema es que tiene que recibir un conjunto de procesos previamente

particionados, de tal manera que dichos procesos se logren poder ejecutar paralelamente en forma eficaz.

Se tiene como una limitación que el sistema actualmente ha sido puesto a prueba con simulaciones de circuitos VLSI (**V**ery **L**arge **S**cale **I**ntegration, integración en escala muy grande) que no permiten determinar de manera cuantitativa la capacidad que se espera. Sin embargo, la implementación realizada asegura que dichas pruebas serán soportadas.

Por otra parte, en este momento el sistema no sabe cuando ha finalizado la ejecución de una aplicación, por tanto para poder finalizarla, se debe de hacer de forma manual cerrando los procesos que se encuentren abiertos.

Además, la cantidad de cómputo en cada proceso debe ser lo bastante significativa como para poder observar la ventaja que se tiene al realizar la ejecución de aplicaciones en forma distribuida en contra de ejecución de aplicaciones en forma centralizadas.

1.11. ESTRUCTURA DE LA TESIS

CAPITULO 1.- Presenta la introducción de esta investigación, donde se muestra el planteamiento del problema, la justificación, objetivos, alcances y limitaciones.
CAPITULO2.- Proporciona el estado del arte en los sistemas que ofrecen laboratorios virtuales para la simulación, y en general sobre los sistemas de Computación en Malla y sus usos, además de mostrar sus conceptos básicos y los de la tecnología .NET Remoting, la cual se implementó para generar la Grid en este proyecto de investigación
CAPITULO 3.- Describe la arquitectura del sistema y la interacción que hay entre sus elementos.
CAPITULO 4.- Muestra los resultados obtenidos además de que se especifican las pruebas realizadas.
CAPITULO 5.- Se presentan las conclusiones y trabajos futuros.

Capítulo 2

Marco Teórico

En este capítulo se da una descripción de los laboratorios virtuales así como de los trabajos relacionados, de los cuales se han tomado ideas para el desarrollo de este proyecto, finalizando con la descripción de las herramientas y tecnologías ocupadas para el desarrollo de este proyecto de investigación.

El contenido de los temas presentados en este capítulo incluye:

- Historia de la Computación en Malla.
- Presentación de trabajos relacionados con la Computación en Malla.
- Características de los laboratorios virtuales.
- Descripción, origen y evolución de los sistemas de Computación en Malla.
- Descripción de las características que conforman la tecnología .NET Remoting.

2.1. HISTORIA DE LA COMPUTACIÓN EN MALLA

La palabra Grid en inglés significa *Malla* y hace referencia a la red eléctrica. Para obtener electricidad, simplemente debe haber conexión a cualquier punto de la *Malla eléctrica* (*power grid*) sin preocuparse por las plantas generadoras que interconectadas brindan esta omnipresencia del servicio; es decir, no se sabe de dónde proviene la electricidad que se utiliza, simplemente se aprovecha.

Este fue el concepto que adoptaron Carl Kesselman e Ian Foster en su libro *The Grid: Blueprint for a new computing infrastructure,* publicado en 1998.

La idea de la Computación en Malla (que incluye cómputo distribuido, programación orientada a objetos y Servicios Web) fue traída por Ian Foster, Carl Kesselman, y Steve Tueque, ampliamente apreciados como los padres del Grid. Ellos condujeron el esfuerzo para crear el Globus Toolkit. No sólo incorporando administración en la computación, sino también administración en el almacenamiento, proporcionando seguridad, movimiento de datos, monitoreo y un kit de herramientas para desarrollar servicios adicionales basados en la misma infraestructura [7].

2.2. PRESENTACIÓN DE TRABAJOS RELACIONADOS CON ESTE PROYECTO DE INVESTIGACIÓN

La programación de flujo de datos es natural y conveniente para describir sistemas de procesamiento de señales digitales, pero en tiempo de ejecución tienen un gran costo. En algunas situaciones los diseñadores no están dispuestos a gastar dinero en recursos de cómputo. Esto es particularmente verdadero cuando la máquina es un circuito integrado programable, tal y como un DSP (Digital Signal Processor, Procesador digital de señal). Sin embargo, en tiempo de ejecución la sobrecarga en las implementaciones LGDF (Large Grain Data Flow, Flujo de Datos de Grano Largo) no requieren más sistemas de procesamiento de señales, ya que cada sistema es en su mayor parte asíncrono. El SDF (Synchronous Data Flow, Flujo de Datos Síncronos) difiere del tradicional flujo de datos en que la cantidad de datos producidos por un nodo de flujo de datos se especifica antes para cada entrada y salida. Esto es equivalente a especificar la prueba relativa evaluada por el sistema de procesamiento

de señales. Esto significa que la planificación de un nodo SDF no necesita terminar en tiempo de ejecución, porque puede terminar en tiempo de compilación (estáticamente), de esta manera se evaporan las sobrecargas en tiempo de ejecución. La prueba evalúa todas las posibles diferencias, las cuales no son verdaderas en los más frecuentes manejadores de datos; en las metodologías de programación para el procesamiento de señales. El flujo de datos síncrono está estrechamente relacionado con los grafos computacionales, un caso especial de redes Petri.

Uno de los trabajos que se relacionan con este proyecto de investigación fue presentado en 1987, donde se desarrolla la teoría necesaria para planificar estáticamente programas SDF en individuales o múltiples procesadores [8]. Una clase estática de algoritmos de planificación es provista válidamente y los algoritmos específicos son dados para la planificación de sistemas SDF en simples o múltiples procesadores.

Hay un amplio uso de los lenguajes de descripción de hardware (HDL) para acelerar el tiempo de comercialización utilizado en el diseño de los sistemas digitales modernos. Para la verificación, los ingenieros pueden simular el hardware con el fin de comprobar su rendimiento y la corrección con la ayuda de un HDL. Sin embargo, la simulación no se puede mantener al ritmo con el crecimiento en tamaño y complejidad de los circuitos y se ha convertido en un cuello de botella del proceso de diseño. La simulación distribuida sobre el HDL en un cluster de supercomputadoras tiene el potencial de proporcionar una solución a este problema.

Otro trabajo relacionado con este trabajo de investigación se presentó en el año de 2003, el cual muestra el diseño y la implementación de un DVS (un área de trabajo orientada a objetos para la distribución de simulaciones Verilog). Verilog es un HDL usado en la industria. DVS es el resultado del clúster Time Warp, originalmente desarrollado para la simulación lógica. El diseño del área de trabajo hace énfasis en la simplicidad, extensibilidad y facilidad para conjuntar experimentos particionados y el balanceo dinámico de cargas de trabajo. (Véase Figura 2.1).

Verilog y VHDL son importantes lenguajes de diseño VLSI (Very Large Scale Integration, integración en escala muy grande). Sin embargo, los esfuerzos de investigación se han centrado hasta el momento en simuladores distribuidos VHDL. Este artículo presenta una descripción de esta investigación, la cual describe el área de trabajo para la simulación distribuida en Verilog.

Escribir un compilador de Verilog representa un importante compromiso. Los autores hicieron uso de Icarus Verilog, un compilador Verilog de código abierto que además es un simulador. También se dieron a la tarea de rediseñar el Time Warp, y se ha utilizado éste como backend para el entorno de simulación [3].

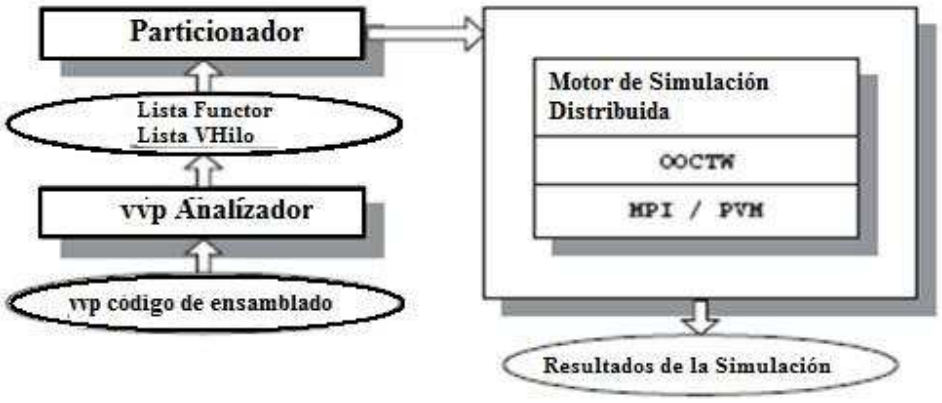

Figura 2.1. Arquitectura del DVS.

Las Mallas computacionalmente acoplan recursos geográficamente distribuidos, por tanto se está volviendo una computación efectiva para resolver problemas de gran escala en la ciencia, la ingeniería y en el comercio. El software para habilitar el Computación en Malla ha sido primordialmente escrito para sistemas operativos Unix, así gravemente limitan la habilidad para utilizar efectivamente los recursos computacionales de las computadoras de escritorio basadas en Windows. La Computación en Malla basada en Windows es particularmente importante desde el punto de vista de las industrias.

Otros trabajos, que si bien no se relacionan con la parte de la simulación de los circuitos digitales, si tienen mucho que ver con el modelo de flujo de datos en las Mallas, el cual ha sido implementado en el sistema descrito en esta tesis, se mencionan a continuación:

Alchemi, un sistema empresarial basado en .NET que provee una maquinaria en tiempo de ejecución y un ambiente de programación para construir Mallas computadoras/escritorio y desarrollar aplicaciones en Malla. Esto permite la composición flexible de aplicaciones soportando un modelo orientado a objetos para programar aplicaciones. El soporte de la plataforma es provisto por medio de una interfaz de servicios Web y de modelos flexibles para la ejecución en nodos Malla dedicados o no dedicados. [9]. Este trabajo data del año 2005.

Alchemi fue concebido con el objetivo de hacer la construcción de la Malla y el desarrollo de software Malla tan fácil como sea posible sin sacrificar la flexibilidad, la adaptabilidad, la fiabilidad y la extensibilidad. (Véase Figura 2.2). Las características que soporta Alchemi son:

- Una agrupación basada en Internet de computadoras de escritorio heterogéneas.
- Nodos individuales para la ejecución, dedicados o no dedicados (voluntarios).
- Un modelo de programación orientado a objetos para aplicaciones Malla.
- Un modelo basado en archivos.
- Una interfaz para los servicios Web que soportan el modelo de trabajo para la interoperabilidad con el middleware.

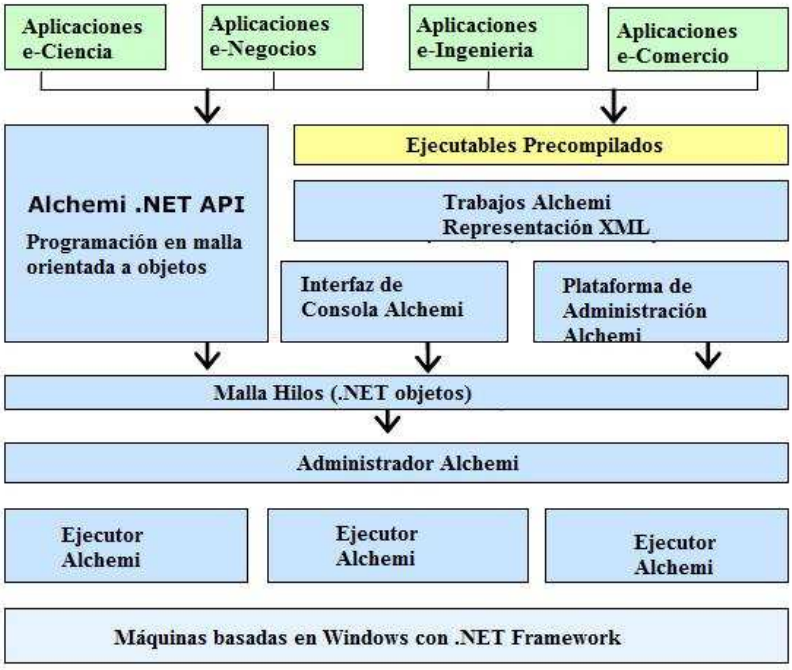

Figura 2.2. Arquitectura de Alchemi.

El trabajo que más incidencia tiene hacia este proyecto es el que se presento en el año 2007, llamado "Un modelo de flujo de datos para sistemas Computación en Malla basados en .NET", está encaminado hacia el soporte avanzado de aplicaciones que contienen múltiples tareas con relaciones entre la dependencia de datos. Algunas aplicaciones de recursos intensivos consisten de

múltiples módulos, los cuales reciben entradas de datos, realizan operaciones y generan resultados. Aplicaciones científicas de esta naturaleza incluyen: simulaciones, minería de datos, y grafos de computación; en muchos casos para esas aplicaciones un módulo de salida de datos se convierte en la entrada de otros módulos. El modelo "flujo de datos de grano grueso" puede usarse para describir tales aplicaciones.

De esta manera ellos usan el modelo de programación de flujo de datos, para crear un grafo de flujo de datos, donde especifican la relación entre la dependencia de datos dentro de aplicaciones distribuidas. Debajo de la interfaz del flujo de datos, usan el motor de flujo de datos para explorar el grafo, para planificar tareas de acuerdo a los recursos distribuidos y el manejo de los problemas difíciles, como ejecuciones escalables, tolerancia a fallos y balanceo de cargas. Dentro de este proceso los usuarios no necesitan preocuparse acerca de los detalles de los procesos, hilos y las comunicaciones explicitas.

Las contribuciones principales de este trabajo son:

- Un simple y poderoso modelo de programación de flujo de datos, el cual soporta la composición de aplicaciones paralelas para desarrollar en ambientes distribuidos
- Una arquitectura y maquinaria en tiempo de ejecución que soporte la planeación de la computación de flujo de datos en ambientes dinámicos y el manejo transparente de fallos.
- Un análisis detallado del modelo de flujo de datos usando dos aplicaciones de ejemplo sobre computadoras de escritorio [10].

En la Figura 2.3 se muestra la arquitectura del sistema descrito anteriormente.

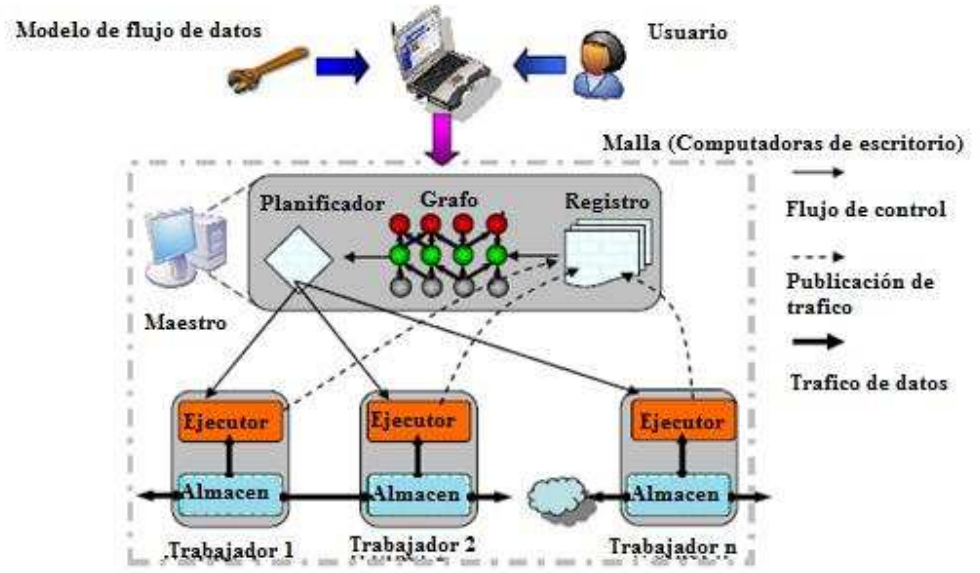

Figura 2.3. Arquitectura del modelo de flujo de datos para sistemas de Computación en Malla basados en .NET.

También en el 2007 fue presentado el diseño de Aneka, una plataforma .NET formada por computadoras de escritorio para la Computación en Malla orientada a servicios, la cual provee: 1) un contenedor de servicio configurable que almacena servicios para descubrir, planificar y balancear cargas de trabajo y 2) una área de trabajo /API (Application Programming Interface, Interfaz de programación de Aplicaciones) para soportar varios modelos de programación en los que se incluyen hilos, procesamiento por lotes, paso de mensajes y flujo de datos. Los usuarios y desarrolladores pueden fácilmente programar diferentes modelos y proveer los servicios del contenedor para correr sus aplicaciones sobre Mallas manejadas por Aneka. De esta manera se presenta la implementación de ambos tanto los esenciales como los servicios avanzados que contiene la plataforma. Evaluando de esta manera el sistema con aplicaciones usando los modelos tarea de Malla y flujo de datos [11].

Aneka fue creada con el fin de proveer un conjunto de servicios que realicen la construcción en Malla y el desarrollo de aplicaciones tan fácil como fuera posible sin sacrificar flexibilidad, escalabilidad, fiabilidad y extensibilidad. (Véase Figura 2.4).

Las mejores características de Aneka son:

- Un contenedor configurable habilitador de servicios, soluciones persistentes, implementaciones de seguridad y protocolos de comunicación.
- Una arquitectura descentralizada permitiendo nodos individuales.
- Múltiples modelos de programación incluyendo el orientado a objetos, hilos en las Mallas, el modelo de programación (abstracción del grano fino), el modelo basado en archivos, basados en tareas (abstracción de grano grueso) y el modelo flujo de datos.
- Múltiples mecanismos de autentificación como seguridad basada en roles, certificados X. 509/GSI proxys y autentificaciones de dominios de Windows.
- Múltiples opciones de persistencia de datos como son RDBMS, ODBMS y archivos XML.
- Interfaces de servicios Web que soportan el modelo de tareas para interoperabilidad con algún intermediario.

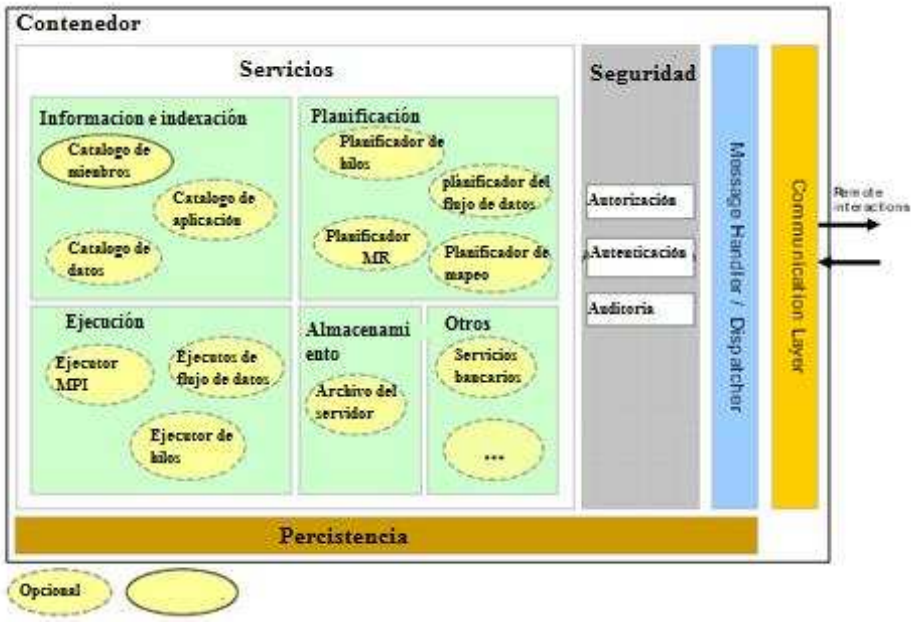

Figura 2.4. Arquitectura de Aneka.

Otro sistema que se ocupa de las tareas anteriormente mencionadas y que data de 2007 llamado "sistema para acelerar la investigación médica basado en el flujo de datos", utiliza un simple lenguaje de scripts, tiene provista la especificación de alto nivel de flujo de trabajo que invoca varias aplicaciones programadas sobre grandes cantidades de trabajo potencialmente, el

motor del Swift está hecho para la ejecución eficiente de esos flujos de trabajo sobre computadoras secuenciales, computadoras paralelas, y/o Mallas distribuidas que obtiene los recursos computacionales de varios sitios. No menos importante, el catálogo de procedencia Swift sigue la pista de todas las acciones realizadas, ocupándose de funciones vitales de contabilidad que tantas veces son causa de dificultades en la computación a gran escala [12]. (Véase Figura 2.5).

Las técnicas antes mencionadas, han alcanzado un gran éxito en aplicaciones basadas en imágenes de investigación neurológica, en las cuales se trata de expandir el alcance y la escala de sus capacidades de cómputo para el estudio de los mecanismos neurológicos del proceso de recuperación de afasia debido a un golpe. Los beneficios de usar el sistema de flujo de trabajo Swift para esta aplicación incluyen grandes reducciones en los tiempos de análisis de datos y en los recursos computacionales. Esto es típicamente requerido en la investigación científica moderna. Este es almacenado automáticamente, en otro caso es una ardua labor manual, además de proveer un acceso transparente de demanda sobre los recursos de la Malla, nuestro flujo de trabajo también exhibe reproducibilidad y procedencia en el rastreo de los datos resultado, eso habilita la colaboración en el proceso de investigación actual, no solo en compartir los resultados, sino en el uso compartido y el aprovechamiento de los procesos de investigación actuales.

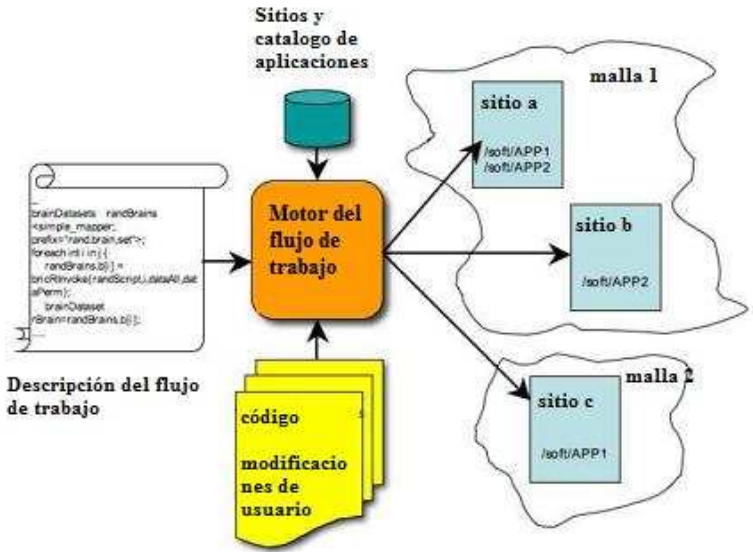

Figura 2.5. Arquitectura del sistema para acelerar la investigación médica basado en el flujo de datos.

2.3. CARACTERÍSTICAS DE LOS LABORATORIOS VIRTUALES

En la Tabla 2.1 se muestran las características que un laboratorio virtual debe tener [2] [28] [29], además se muestran varios ejemplos en los se observan si cumplen con las características.

Tabla 2.1. Características de los laboratorios virtuales.

CARACTERISTICAS DE LOS LABORATORIOS VIRTUALES	PSPICE	LABVIEW	ORCAD	WEBTOP	TOOLWIRE
-Permite tener un ambiente lo mas similar posible a los laboratorios tradicionales	**NO**	**SI**	**NO**	**NO**	**SI**
-Permite realizar simulación de fenómenos, modelos físicos, conceptos abstractos.	**SI**	**SI**	**SI, en nuevas versiones integra a Pspice**	**NO**	**SI**
-Permite tener una visión más intuitiva de aquellos fenómenos que en su realización manual no soportan suficiente claridad gráfica, (Análisis de datos).	**SI**	**SI**	**SI**	**NO**	**SI**
-Acerca y facilita la experimentación a un mayor número de personas.	**SI**	**SI**	**SI, puede trabajar en red**	**NO**	**SI**
-Permite realizar experimentación iterativa.	**SI**	**SI**	**SI**	**NO**	**SI**

2.4. LABORATORIOS VIRTUALES: VENTAJAS

Un Laboratorio Virtual (LV) es un sistema computacional que pretende aproximar el ambiente de un Laboratorio Tradicional (LT). Los experimentos se realizan paso a paso, siguiendo un procedimiento similar al de un LT: se visualizan instrumentos y fenómenos mediante objetos dinámicos (applets de Java o Flash, cgi-bin, javascripts,...), imágenes o animaciones. A continuación, se destacan algunas ventajas importantes de los LV [2]:

- Acerca y facilita a un mayor número de usuarios la realización de experiencias, aunque usuario y laboratorio no coincidan en el espacio. Las personas acceden a los equipos del laboratorio a través de un navegador, pudiendo experimentar sin riesgo alguno, y, además, se flexibiliza el horario de prácticas y evita la saturación por el traslape con otras actividades.
- Reducen el costo del montaje y mantenimiento de los LT, siendo una alternativa barata y eficiente, donde las personas simulan los fenómenos a estudiar como si los observara en el LT.
- Es una herramienta de auto aprendizaje, donde el usuario altera las variables de entrada, configura nuevos experimentos, aprende el manejo de instrumentos, personaliza el experimento, etc. La simulación en el LV, permite obtener una visión más intuitiva de aquellos fenómenos que en su realización manual no aportan suficiente claridad gráfica. El uso de LV da lugar a cambios fundamentales en el proceso habitual de enseñanza, en el que se suele comenzar por el modelo matemático. La simulación interactiva de forma aislada posee poco valor didáctico, ésta debe ser embebida dentro de un conjunto de elementos multimedia que guíen al usuario eficazmente en el proceso de aprendizaje. Se trata de utilizar la capacidad de procesamiento y cálculo del ordenador, incrementando la diversidad didáctica, como complemento eficaz de las metodologías más convencionales.
- Las personas aprenden mediante prueba y error, sin miedo a sufrir o provocar un accidente, sin avergonzarse de realizar varias veces la misma práctica, ya que pueden repetirlas sin límite; sin temor a dañar alguna herramienta o equipo. Pueden asistir al laboratorio cuando ellos quieran, y elegir las áreas del laboratorio más significativas para realizar prácticas sobre su trabajo.

2.5. LABORATORIOS VIRTUALES: DESVENTAJAS

- El LV no puede sustituir la experiencia práctica altamente enriquecedora del LT. Ha de ser una herramienta complementaria para formar a la persona y obtener un mayor rendimiento.
- El alumno no utiliza elementos reales en el LV, lo que provoca una pérdida parcial de la visión de la realidad. Además, no siempre se dispone de la simulación adecuada para el tema que se desea trabajar. En ocasiones ciertas simulaciones pueden requerir una cantidad de recursos bastante grande ya que realizan tareas bastante intensivas en cuanto a cómputo, como para poder ser ejecutadas en una sola computadora aislada.

En base a esta última desventaja surge una pregunta: ¿Las simulaciones pueden ser realizadas bajo un esquema de cómputo distribuido?

Todas aquellas simulaciones que puedan ser paralelizables son adecuadas para trabajar bajo el esquema distribuido, donde se tienen varias alternativas para ello:

- **Computación distribuida:** Es una red de ordenadores donde los recursos informáticos son compartidos con todos los otros ordenadores en el sistema. La potencia de procesamiento, la memoria y el almacenamiento de datos, son recursos de la comunidad donde los usuarios autorizados pueden entrar y realizar ciertas tareas.
- **Multi - hilo:** La habilidad de un sistema operativo para ejecutar diferentes partes de un programa, llamados hilos, simultáneamente. El programador debe diseñar cuidadosamente el programa de tal manera que todos los hilos pueden ejecutar al mismo tiempo, sin interferir unos con otros.
- **Modelo de paso de mensajes:** En los modelos de paso de mensajes, los procesos están activos y la información es intercambiada usando un mecanismo de paso de mensajes de una forma u otra. Aunque el mecanismo puede resultar poderoso, fuerza al programador a considerar exactamente donde una comunicación muy costosa debe tener lugar.
- **Computación en Malla**: Se enmarca dentro de la tecnología de computación distribuida. Su funcionalidad principal es la de compartir potencia computacional. Se basa en el aprovechamiento de los ciclos de procesamiento no utilizados por los millones de ordenadores conectados en la Red. De esta forma se consigue que puedan resolver de forma distribuida tareas que son demasiado intensivas para ser resueltas por una máquina aislada.

2.6. TIPOS DE SERVICIOS QUE OFRECE UN LABORATORIO VIRTUAL

- **Simulación digital.** Servicio que puede ser paralelizable.
- **Simulación analógica.** Servicio que puede ser paralelizable.
- **Compilación.** Servicio que no puede ser paralelizable.
- **Auto ruteo.** Servicio que puede ser paralelizable.
- **Herramientas**. Osciloscopio, generadores de ondas, gestores de cables, gestores de chips, entre otros. No pueden ser paralelizables.

2.7. COMPUTACIÓN EN MALLA

2.7.1. CONCEPTO DE COMPUTACIÓN EN MALLA

La funcionalidad principal de la Computación en Malla es compartir potencia computacional [13]. Desde el momento en el que los primeros ordenadores comenzaron a conectarse a Internet, surgió la idea de unir la potencia inutilizada de cada uno para abordar problemas a los que sólo podían enfrentarse las supercomputadoras pertenecientes a organizaciones gubernamentales, universidades o grandes multinacionales.

La tecnología que hace esto posible se llama Grid Computing (Computación en Malla) y se basa en el aprovechamiento de los ciclos de procesamiento no utilizados por los millones de ordenadores conectados a la Red (Véase Figura 2.6). De esta forma se consigue que puedan resolver tareas que son demasiado intensivas para ser resueltas por una máquina aislada.

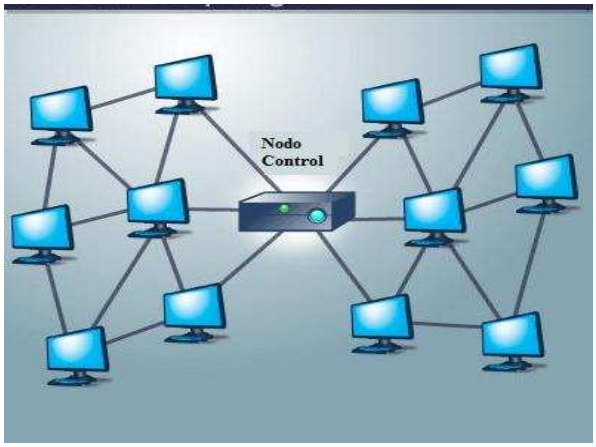

Figura 2.6. Representación de la Computación en Malla.

2.7.2. VENTAJAS QUE OFRECE LA COMPUTACIÓN EN MALLA

Los principales beneficios son:

- Ofrecer flexibilidad para llenar las necesidades cambiantes del negocio.
- Brindar alta calidad a menor costo.
- Facilitar el pronto retorno de las inversiones.
- No necesitar de toda una nueva infraestructura para que funcione.
- Facilitar poder de computación / precio barato.

- Brindar el poder de un supercomputador.
- Utilizar software gratuito y usar código fuente abierto.
- No precisar hardware adicional, para posibilitar el incremento de la potencia de cómputo.
- Brindar transparencia para el usuario que participa en la Malla.

2.7.3. INTRODUCCIÓN A LA COMPUTACIÓN EN MALLA

La idea de la Malla está enfocada fundamentalmente en el acceso remoto a recursos computacionales y pretende ser un paradigma de desarrollo no centrado en una tecnología concreta [14]. La evolución de la Computación en Malla se refleja en el avance de la estandarización de esta tecnología (el estándar de Globus Project es el estándar de facto) donde se encuentra definida la arquitectura de la Malla, los niveles de acceso, los requisitos, los servicios, etc.

La Computación en Malla se enmarca dentro de la tecnología de computación distribuida englobando conceptos como: sistemas operativos distribuidos, programación multiprocesador, computación paralela, redes de computadoras, seguridad, bases de datos, etc. De alguna manera el concepto de Computación en Malla da una unidad conceptual a estos problemas de manera que todos ellos puedan verse desde una perspectiva en Malla.

La Computación en Malla es más que una idea ambiciosa, ya que no sólo se trata de compartir ciclos de CPU para realizar cálculos complejos sino que se busca la creación de una infraestructura distribuida. Esta ardua tarea involucra labores de definición de la arquitectura general, de interconexión de diferentes redes, de definición de estándares, de desarrollo de procedimientos para la construcción de aplicaciones, etc. [15]. La Malla es una idea que promete revolucionar el mundo de la computación y el cómo se desarrollan las aplicaciones actualmente [13].

2.7.4. ARQUITECTURA DE LA COMPUTACIÓN EN MALLA

Principalmente, la arquitectura propuesta es una arquitectura de protocolos que definen los mecanismos básicos que permiten a los usuarios y a los recursos negociar, establecer, gestionar y explotar la compartición de recursos. Una arquitectura abierta basada en un estándar facilita la extensibilidad, la interoperatibilidad, la portabilidad y la compartición de código. De esta manera la estandarización de los protocolos permitirá regular los servicios y mejorar las capacidades de la

Malla. En el nivel de infraestructura se encuentran los recursos computacionales, como son los ordenadores, los clusters, las supercomputadoras, los sistemas de almacenamiento en red, las bases de datos, etc. También se incluyen en este nivel la infraestructura de la red y sus mecanismos de gestión y control. En la terminología de Malla, la infraestructura se denomina la fábrica y suministra los componentes que serán compartidos. (Véase Figura 2.7).

El nivel de conectividad incluye los protocolos de comunicación y seguridad que permiten a los recursos computacionales comunicarse. Entre estos protocolos se encuentran: la pila de protocolos TCP/IP, el protocolo SSL, Certificados X.509.

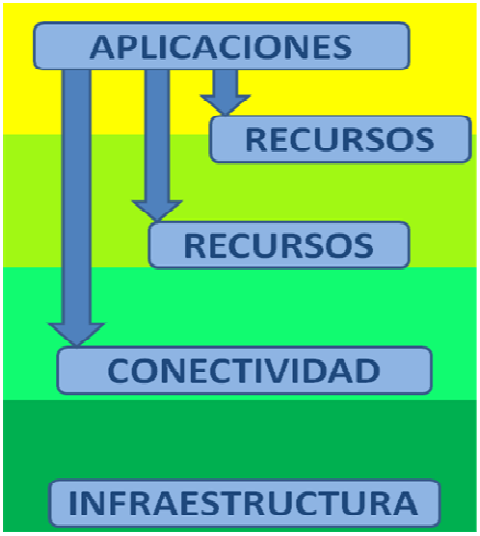

Figura 2.7. Arquitectura de la Computación en Malla.

El nivel de recurso se centra en la gestión de un único recurso y permite tener información y control sobre él mismo. En este nivel se encuentran los protocolos que permiten obtener la información de un recurso: las características técnicas, la carga actual, el precio, etc. También se encuentran los protocolos que permiten el control del recurso: el acceso al mismo, el arranque de procesos, la gestión, la parada, la monitorización, la contabilidad de uso y la auditoria del recurso [16].

La capa de recursos engloba todos los servicios que permiten gestionar un conjunto de recursos. Se encuentran los servicios de directorio, que permiten localizar los recursos que son de interés; los planificadores distribuidos, que permiten asignar las tareas a cada recurso; la monitorización y diagnóstico de la ejecución de las distintas tareas en que se distribuyen la ejecución de una

aplicación; la contabilidad, que permite calcular el coste de la utilización de varios recursos heterogéneos, y el acceso a datos distribuidos, que gestiona la replicación de datos.

El servicio planificador distribuido es una de las aplicaciones más complejas de un desarrollo en Malla, ya que existen planificadores distintos: El planificador que maximiza el uso de los recursos, el planificador de la aplicación que divide la aplicación en tareas, el planificador que asigna los recursos para su ejecución y vigila el desarrollo de los mismos. Los dos primeros verifican la eficiencia del sistema Malla, mientras que el tercero verifica la eficiencia de la aplicación.

El último nivel, el de aplicación, se centra en la definición de protocolos, que permiten a las aplicaciones el acceso a la infraestructura de la Malla a través de las distintas capas. Según el tipo de aplicación puede ser necesario conectarse a las distintas capas o acceder directamente a una de ellas, incluso directamente a la infraestructura de trabajos (planificador de trabajo) que intenta maximizar la cantidad de trabajo realizado (trabajos por unidad de tiempo).

2.7.5. Lo que la Computación en Malla Puede Hacer

2.7.5.1. Aprovechando los Recursos que no Siempre se Usan

El uso más fácil de la Computación en Malla es ejecutar una aplicación existente en una máquina diferente [17] [15] [13] [18].

La máquina en que la aplicación normalmente se ejecuta podría estar inusualmente ocupada debido a un pico inusual de actividad. El trabajo en cuestión podría ejecutarse en otra parte en una máquina ociosa en la Malla.

Hay dos requisitos previos a considerar. Primero, la aplicación debe ser ejecutable remotamente. Segundo, la máquina remota debe encontrar cualquier hardware especial, software, o requerimientos de recursos impuestos por la aplicación. Por ejemplo, un trabajo en lotes (batch) que consume una cantidad significativa de tiempo, procesando un conjunto de datos de entrada para producir un conjunto de resultados, es quizás el uso más ideal y simple para una Malla. Si las cantidades de entradas y salidas son grandes, más análisis y planeación podrían requerirse para usar eficazmente la Malla para tal trabajo.

2.7.5.2. CAPACIDAD DE CPU PARALELA

El potencial para la capacidad de CPU paralela masiva es uno de los rasgos más atractivos de una Malla. Además de necesidades científicas puras, tal poder de cómputo está conduciendo a una nueva evolución en las industrias como el campo bio-médico, planeación financiera, exploración petrolera, etc. El atributo común entre tales usos es que las aplicaciones se han escrito para usar algoritmos que pueden dividirse independientemente en partes de ejecución.

Una aplicación de Malla intensiva de CPU puede pensarse como muchos sub-trabajos más pequeños, cada uno ejecutándose en una máquina diferente en la Malla.

2.7.5.3. EL ACCESO A LOS RECURSOS ADICIONALES

Los recursos adicionales pueden proporcionarse en número y capacidad variable. Por ejemplo, si un usuario necesita aumentar su ancho de banda total a Internet para implementar un mecanismo de búsqueda de minería de datos, el trabajo podría dividirse entre máquinas de la Malla que tienen conexiones independientes a Internet.

2.7.5.4. BALANCEO DE RECURSOS

Para aplicaciones habilitadas, la Malla puede ofrecer un efectivo balanceo de recursos mediante la planificación de trabajos de Malla en máquinas con poca utilización.

Esta facilidad puede mejorar invalorablemente el manejo de picos de carga de actividad en sectores de una organización más grande. Esto puede pasar de dos maneras:

- Un pico inesperado puede ser conducido a máquinas relativamente ociosas en la Malla.
- Si la Malla ya se utiliza totalmente, el trabajo de prioridad más baja que se realiza en la Malla debe ser suspendido temporalmente o incluso cancelado y realizado posteriormente para dejar lugar a un trabajo de prioridad mayor.

Sin una infraestructura de Malla, tales decisiones de equilibrio serían difíciles de priorizar y ejecutar [19].

2.8. ÁREA DE TRABAJO DE .NET REMOTING

2.8.1. INTRODUCCIÓN

El área de trabajo de .NET Remoting, (Véase Figura 2.8), permite crear fácilmente aplicaciones ampliamente distribuidas, tanto si los componentes de las aplicaciones están todos en un equipo, como si están repartidos por el mundo. Se pueden crear aplicaciones de cliente que utilicen objetos en otros procesos del mismo equipo o en cualquier otro equipo disponible en la red. También se puede utilizar .NET Remoting para comunicarse con otros dominios de aplicación en el mismo proceso. También .NET Remoting proporciona un enfoque abstracto en la comunicación entre procesos que separa el objeto utilizado de forma remota de un dominio de aplicación de cliente o servidor específico, y de un mecanismo específico de comunicación. Por lo tanto, se trata de un sistema flexible y fácilmente personalizable. Se puede reemplazar un protocolo de comunicación con otro o un formato de serialización con otro, sin tener que recompilar el cliente ni el servidor. Además, el sistema de interacción remota no presupone ningún modelo de aplicación en particular. Se puede comunicar desde una aplicación Web, una aplicación de consola, un servicio de Windows o desde casi cualquier aplicación que se desee utilizar. Los servidores de interacción remota también pueden ser cualquier tipo de dominio de aplicación. Cualquier aplicación puede albergar objetos de interacción remota y proporcionar sus servicios a cualquier cliente en su equipo o red [20].

Figura 2.8. Elementos que componen el área de trabajo de .NET Remoting

2.8.2. ARQUITECTURA DEL ÁREA DE TRABAJO .NET REMOTING

La verdadera ventaja del sistema de interacción remota es su capacidad para permitir la comunicación entre objetos pertenecientes a dominios de aplicación o a procesos distintos mediante diferentes protocolos de transporte, formatos de serialización, esquemas de duración de objetos y modos de creación de objetos (Véase Figura 2.9). Además, la interacción remota permite intervenir en prácticamente todas las fases del proceso de comunicación, sea cual sea la razón [21].

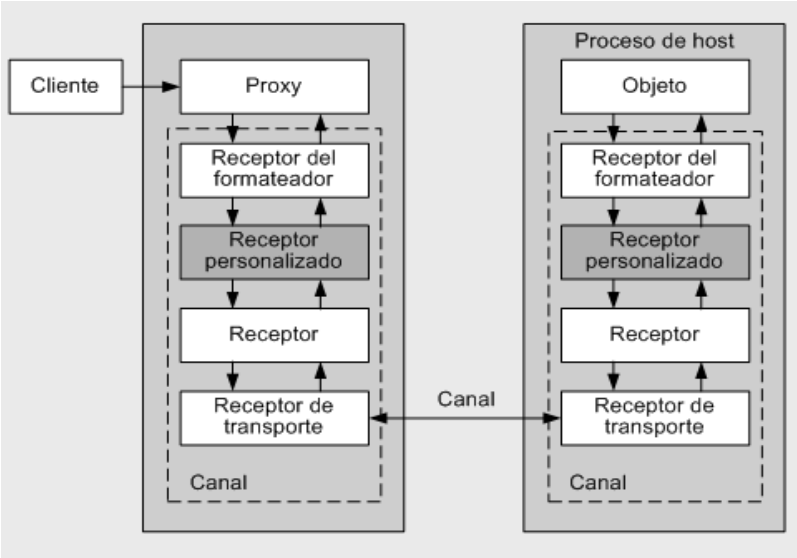

Figura 2.9. Arquitectura de la área de trabajo .NET Remoting

2.8.2.1. COPIAS Y REFERENCIAS

La comunicación entre procesos requiere un objeto servidor cuya funcionalidad esté a disposición de los llamadores fuera de su proceso, un cliente que realice llamadas al objeto servidor y un mecanismo de transporte que lleve las llamadas de un extremo a otro. Las direcciones de los métodos del servidor son lógicas y funcionan correctamente en un proceso, pero no funcionan en otro proceso de cliente. Para solucionar este problema, el cliente puede llamar a un objeto servidor realizando una copia de todo el objeto y pasándola al proceso de cliente, donde se pueden invocar directamente los métodos de la copia.

2.8.2.2. ARQUITECTURA SIMPLIFICADA DE INTERACCIÓN REMOTA

La arquitectura de interacción remota proporciona al programador un procedimiento aún más sencillo. Si configura correctamente el cliente, sólo tiene que crear una nueva instancia del objeto remoto mediante **new** (o la función de creación de instancias del lenguaje de programación administrado que utilice). Su cliente recibe una referencia al objeto de servidor, lo que le permite llamar a sus métodos como si el objeto estuviera en su proceso en lugar de estar ejecutándose en otro equipo. El sistema de interacción remota utiliza objetos proxy para dar la impresión de que el objeto del servidor se encuentra en el proceso del cliente. Los objetos proxy son objetos complementarios, que se presentan como si fueran otro objeto. Cuando un cliente crea una instancia del tipo remoto, la infraestructura de interacción remota crea un objeto proxy que, para su cliente, tiene exactamente la misma apariencia que el tipo remoto. Su cliente llama a un método en ese objeto proxy y el sistema de interacción remota recibe la llamada, la dirige hacia el proceso del servidor, invoca al objeto de servidor y envía el valor devuelto al objeto proxy del cliente, que a su vez devuelve el resultado al cliente. En el sistema .NET Remoting, la combinación de tecnologías subyacentes necesarias para abrir una conexión de red y utilizar un determinado protocolo para enviar los bytes a la aplicación receptora se representa como un canal de transporte.

2.8.2.3. DISEÑO COMPLETO DE UN SISTEMA DE INTERACCIÓN REMOTA

Para poder citar un ejemplo tenemos que imaginar que deseamos utilizar la funcionalidad de .NET Remoting para realizar la comunicación hacia otro equipo. (Véase Figura 2.10).

Un cliente se limita a crear una nueva instancia de la clase de servidor. El sistema de interacción remota crea un objeto proxy que representa a la clase y devuelve al objeto del cliente una referencia al objeto proxy. Cuando un cliente llama a un método, la infraestructura de interacción remota controla la llamada, comprueba el tipo de información y dirige la llamada por el canal hacia el proceso del servidor. Un canal a la escucha detecta la solicitud y la reenvía al sistema de interacción remota del servidor, que a su vez busca (o crea, si es necesario) y llama al objeto solicitado. A continuación el proceso se invierte: el sistema de interacción remota del servidor incluye la respuesta en un mensaje que el canal del servidor envía al canal del cliente. Por último, el sistema de interacción remota del cliente devuelve el resultado de la llamada al objeto del cliente a través del objeto proxy.

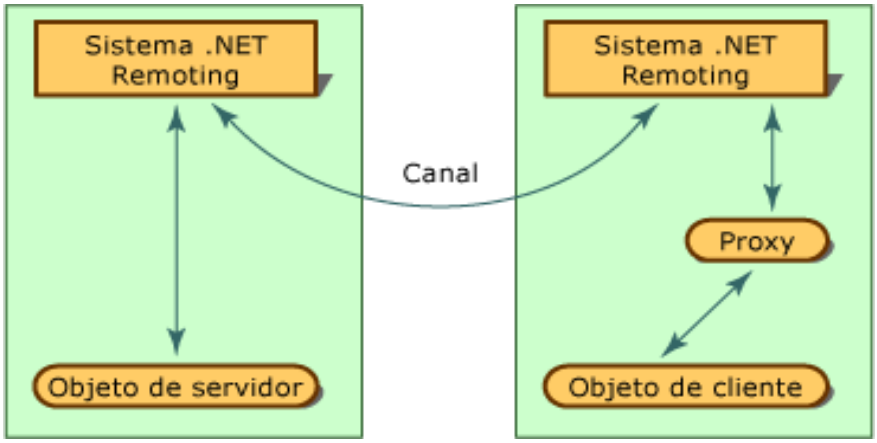

Figura 2.10. Proceso de interacción remota

2.8.3. CONVERTIR OBJETOS EN OBJETOS UTILIZABLES DE FORMA REMOTA

2.8.3.1. PROCESOS Y DOMINIOS DE APLICACIÓN

Los sistemas operativos y los entornos de motores de tiempo de ejecución modernos necesitan proteger cada aplicación frente a los errores de las demás aplicaciones. Este mecanismo de protección se implementa mediante el uso de procesos y dominios de aplicación [22].

2.8.3.2. PROCESOS

Para proteger a unas aplicaciones de otras, en los sistemas operativos Microsoft Windows se ejecutan cada una en su propio proceso. Si se produce un error en una aplicación por algún motivo, sólo se ve afectado ese proceso, mientras que las aplicaciones de otros procesos siguen funcionando. Naturalmente, debido a que las direcciones de memoria en un proceso no tienen sentido en ningún otro, puede resultar un tanto complejo llamar a las funciones de un proceso desde otro. Cálculo de referencias es el término asignado a los eventos que se producen cuando una llamada y sus argumentos se empaquetan en un proceso y se desempaquetan en otro, de manera que una llamada que atraviese el límite de un proceso pueda realizarse.

2.8.3.3. DOMINIOS DE APLICACIÓN

En el entorno administrado, los dominios de aplicación, que se pueden considerar como procesos lógicos, y los contextos proporcionan aislamiento y seguridad a un costo menor y con una capacidad mayor para escalar correctamente que un proceso de sistema operativo, gracias, entre otros factores, al hecho de que el código administrado dispone de seguridad de tipos verificable. Toda aplicación administrada se ejecuta en un dominio de aplicación, tanto si otra aplicación inicia un dominio en su lugar como si el entorno host inicia uno por ella. .NET Remoting facilita la infraestructura para comunicarse entre dominios de aplicación de una manera sencilla, protegida por las tecnologías de seguridad.

2.8.4. PERIODOS Y ACTIVACIÓN DE OBJETOS REMOTOS

En el momento de crear un objeto remoto, se tiene que saber cuándo y cómo se crea y se inicializa un nuevo objeto; esto es, cómo se activa. Dado que el sistema de interacción remota siempre necesita saber qué tipo de activación es necesario para poder poner los objetos a disposición de los clientes [23].

2.8.4.1. ACTIVACIÓN DE OBJETOS REMOTOS

Existen dos tipos de activación para los objetos de cálculo por referencia:

- Activación por el servidor.

  Los objetos activados por el servidor los crea el servidor únicamente cuando son necesarios. No se crean al crear el proxy de cliente llamando a **new**, sino cuando el cliente invoca al primer método de dicho proxy.

  Se utiliza la enumeración WellKnownObjectMode para configurar los objetos activados en el servidor como objetos **Singleton** o **SingleCall**. Los objetos **Singleton** son aquellos para los que siempre habrá una sola instancia, independientemente de cuántos clientes haya para ese objeto y de cuáles tengan una duración predeterminada (el cliente puede usar el sistema de concesión del período para participar en la duración de las instancias **Singleton**). Cuando se configura un objeto como **SingleCall**, el sistema crea un objeto nuevo por cada llamada a un método de un cliente. Dado que un cliente recibe una referencia a una nueva instancia

con cada llamada, los tipos **SingleCall** no participan en el sistema de concesión del período de duración.

- Activación por el cliente.

  Los objetos activados en el cliente se crean en el servidor cuando el cliente llama a **new**. El propio cliente, usando el sistema de concesión del período de duración, puede participar en la duración de estas instancias [24].

2.8.4.2. PERÍODOS

Los objetos MBR (Marshal-By-Reference, cálculo por referencia) no residen en la memoria eternamente, tanto si son objetos **Singleton** activados en el servidor como si son objetos activados en el cliente. En cambio, a no ser que el tipo reemplace a MarshalByRefObject.InitializeLifetimeService para controlar sus propias directivas referentes a la duración, cada objeto MBR tiene una duración controlada por una combinación de concesiones, un administrador de concesiones y una serie de patrocinadores (en este caso, la duración de un objeto MBR equivale al tiempo total que el objeto permanece activo en la memoria.). Una concesión es el período que un determinado objeto está activo en la memoria antes de que el sistema .NET Remoting comience el proceso para eliminarlo y recuperar la memoria. El administrador de concesiones del dominio de aplicación de servidor es el objeto que determina cuándo el objeto remoto debe ser marcado por el recolector de elementos no utilizados. Un patrocinador es un objeto que solicita una nueva concesión para un determinado objeto, para lo que se registra él mismo en el administrador de concesiones.

Puesto que la vida útil de un objeto remoto es independiente a la de sus clientes, la concesión para un objeto sencillo o pequeño puede ser muy larga, la pueden utilizar varios clientes y la puede renovar periódicamente un administrador o un cliente. Este enfoque utiliza las concesiones de manera eficaz, porque se necesita muy poco tráfico en la red para la recolección distribuida de elementos no utilizados. Sin embargo, los objetos remotos que utilizan recursos escasos pueden tener una concesión para un período de duración breve, que el cliente renueva frecuentemente a intervalos cortos. Cuando todos los clientes han terminado con el objeto remoto, el sistema .NET Remoting elimina rápidamente el objeto. Con esta táctica, en lugar de aumentar el tráfico en la red, se utilizan de una manera más eficaz los recursos del servidor [25].

2.8.5. CANALES

Los canales son objetos que transportan mensajes de una aplicación a otra a través de los límites de interacción remota, tanto de un dominio de aplicación a otro, como de un proceso a otro o de un equipo a otro. Un canal puede escuchar los mensajes entrantes en un extremo, enviar los mensajes salientes a otro extremo o ambas acciones [26].

En el cliente, los mensajes se pasan a la cadena de receptores de canal del cliente después de que hayan recorrido la cadena de contextos del cliente. El primer receptor de canal suele ser un receptor de formato que serializa el mensaje en una secuencia que a continuación, pasa por la cadena de receptores de canal hasta el receptor de transporte del cliente. El receptor de transporte del cliente escribe entonces la secuencia en la conexión.

Los canales deben implementar la interfaz IChannel, que proporciona propiedades informativas como ChannelName y ChannelPriority. Los canales diseñados para estar atentos a la llegada de un determinado protocolo a un determinado puerto implementan IChannelReceiver mientras que los canales diseñados para enviar información implementan IChannelSender. Tanto el objeto **TcpChannel** como el objeto **HttpChannel** implementan estas dos interfaces, por lo que se pueden utilizar para enviar o recibir información.

En el servidor, el receptor de transporte del servidor lee las solicitudes de la conexión y pasa la secuencia de solicitud a la cadena de receptores de canal del servidor. El receptor de formato del servidor situado al final de esta cadena des serializa la solicitud en un mensaje. A continuación, lo pasa a la infraestructura de interacción remota.

Capítulo 3

Descripción de la Arquitectura del Sistema

Se detalla la arquitectura del sistema donde se describen cada uno de los elementos que la conforman, resaltando a aquellos que realizan el funcionamiento de Computación en Malla y a los que desarrollan el modelo de paso de mensajes en forma punto a punto para solventar la dependencia de datos, así también la interacción que realizan entre sí.

El contenido de los temas presentados en este capítulo incluye:

- Descripción del marco de trabajo.
- Descripción de la arquitectura del sistema.

3.1. DESCRIPCIÓN DEL MARCO DE TRABAJO

Cada día va en aumento la creación de laboratorios virtuales que están siendo montados sobre Internet con la premisa de ofrecer un conjunto de herramientas de Automatización y Diseño Electrónico (EDA) (Véase Figura 3.1). Los desarrolladores de estos laboratorios están combinado interfaces y herramientas amigables para que por medio de éstas, los usuarios puedan tener a su disposición todos los medios que tendrían en un laboratorio tradicional, pero con la gran ventaja de que puedan ser utilizados desde cualquier parte del mundo en el horario que se adecue a cada uno de los usuarios. Cada día existen más trabajos que pretenden ofrecer laboratorios para que se atiendan las necesidades de cada usuario, brindándoles como un plus, una forma de experimentación más gráfica e ilustrativa.

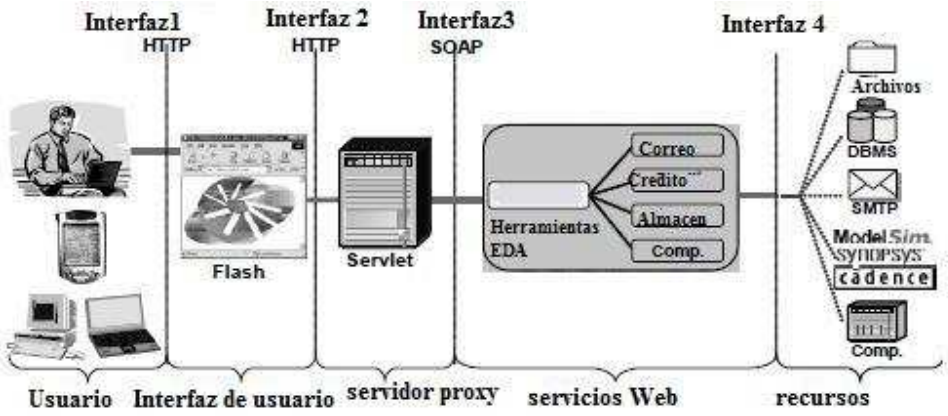

Figura 3.1. Entornos y herramientas que ofrecen los laboratorios virtuales.

Si bien, va en aumento el trabajo que está encaminado al desarrollo de laboratorios virtuales, muy pocos de éstos, están procurando ofrecer la capacidad de realizar cómputo distribuido para mejorar el rendimiento a la hora de ejecutar aplicaciones que requieren una gran cantidad de recursos informáticos y una gran capacidad de poder de cómputo. Por este motivo este proyecto de investigación está enfocado a brindar un sistema que les permita a los laboratorios virtuales ejecutar de forma distribuida todas aquellas tareas que son demasiado intensivas como para ser resueltas por una sola maquina aislada, utilizando a la Computación en Malla para ello (Véase Figura 3.2).

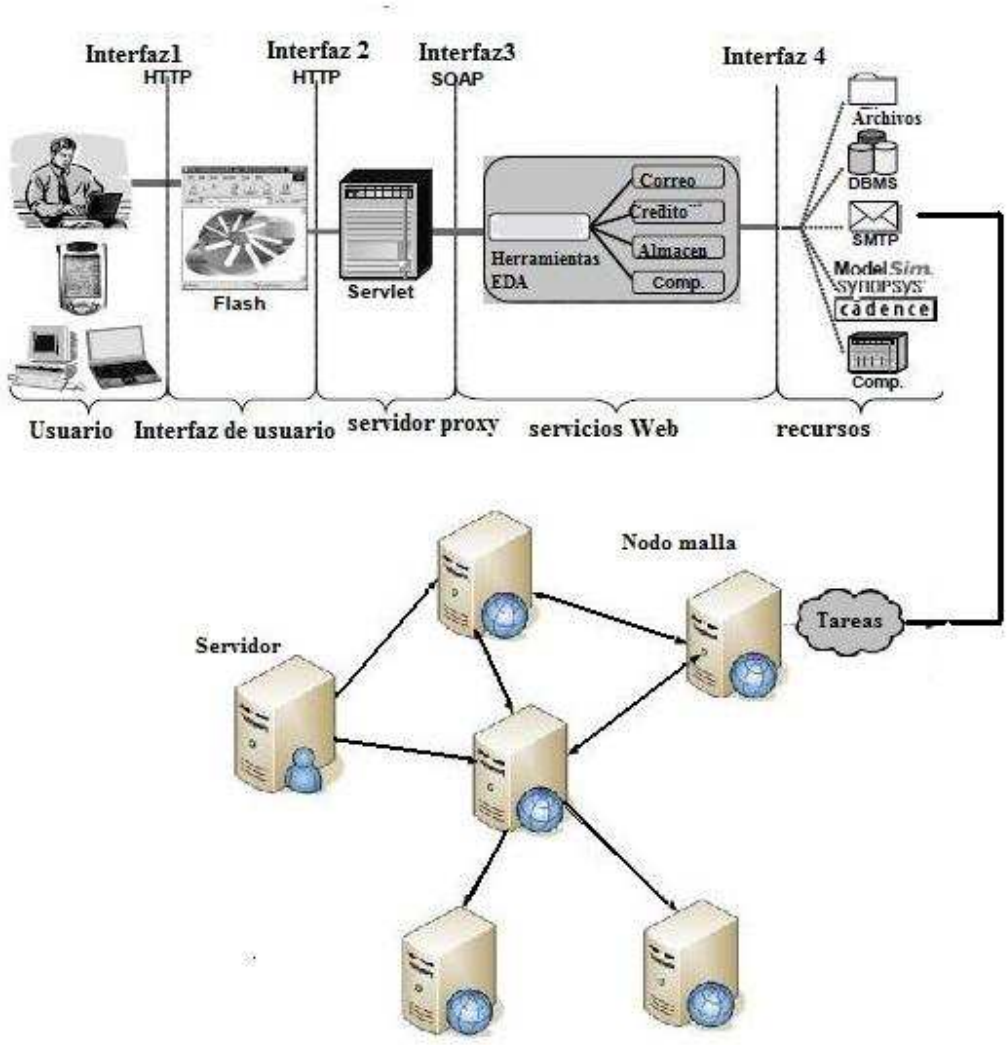

Figura 3.2. Integración del sistema de Computación en Malla a los laboratorios virtuales.

3.2. ARQUITECTURA DEL SISTEMA

En la Figura 3.3, se muestra la arquitectura que conforma al sistema, esta consta de tres elementos principales: 1) Un sistema que crea la plataforma estática del sistema, así como de la presentación de resultados al usuario, estas tareas se realizan cada vez que una nueva aplicación va a ser ejecutada en el sistema, 2) Un maestro que se encarga de realizar el balanceo de cargas de trabajo y la distribución de procesos que conforman a la aplicación que se pretende ejecutar en forma distribuida, 3) De múltiples unidades de ejecución, las

cuales a partir de este momento serán llamadas trabajadores, si bien, éstos se encargan de realizar la ejecución de los procesos y de solventar la dependencia de datos que exista entre ellos.

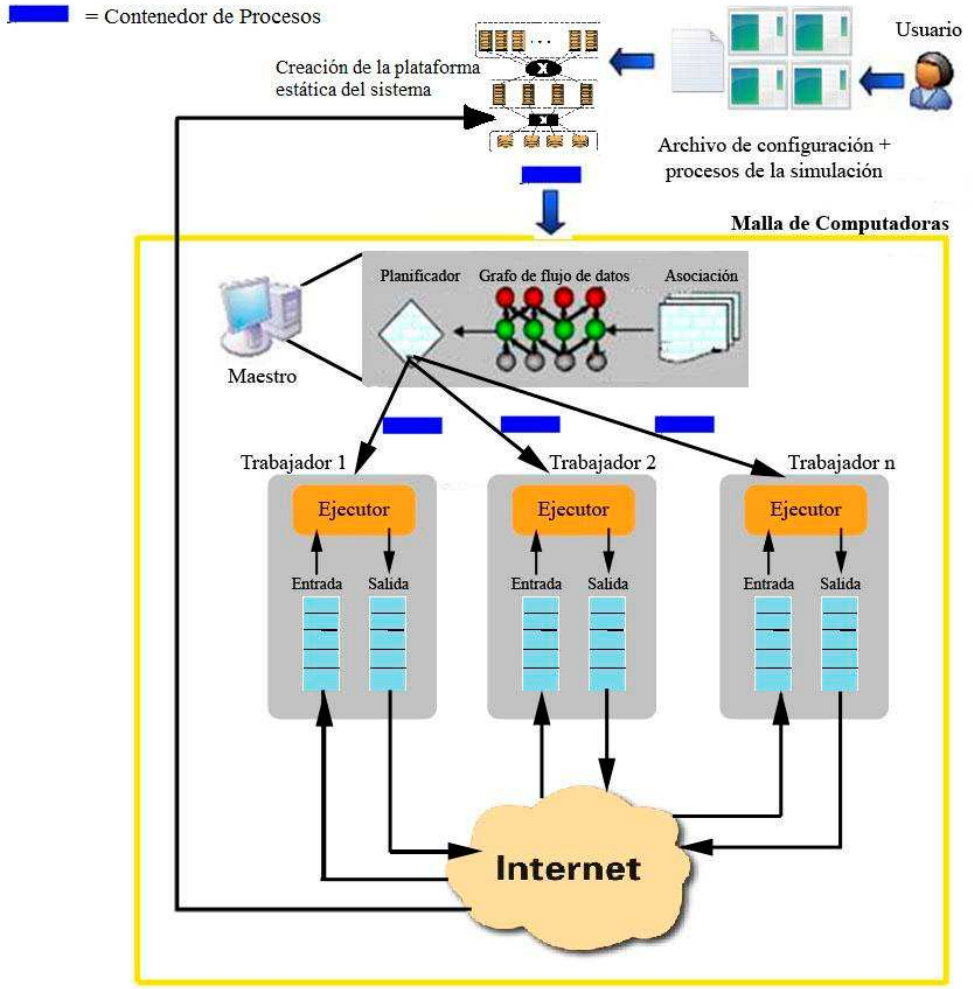

Figura 3.3. Arquitectura del sistema.

3.3. SISTEMA PARA LA CREACIÓN DE LA PLATAFORMA ESTÁTICA

Esta parte de la arquitectura comprende tres elementos, los cuales son: el archivo de configuración de la aplicación que va a ser ejecutada en el sistema más los procesos que la conforman, el sistema para la creación de la plataforma estática y por último un contenedor de procesos. Estos elementos se describen a continuación.

3.3.1. ARCHIVO DE CONFIGURACIÓN

La estructura del archivo, describe en cada una de sus líneas al grafo de dependencias de los datos, lo que quiere decir, que procesos se comunican entre sí (Véase Figura 3.4). La estructura está definida de la siguiente manera:

En la fila número 1 se indica el PID (Process Identification, Identificador Único de Proceso) con el cual se identificarán a todos los procesos de una aplicación en concreto que se vaya a ejecutar en el sistema.

A partir de la fila número 2, se indica información perteneciente a cada uno de los procesos, cada campo está separado por comas; donde primeramente se indica el nombre del proceso, posteriormente muestra su peso (requerimientos del proceso), después indica las dependencias entre de datos que tiene con otros procesos. Esto es, el nombre o los nombres de los procesos que necesitarán los resultados que éste produzca, a continuación el número total de comunicaciones que habrá entre esos procesos, seguido de esto, muestra la posición de la cola de almacenamiento del proceso, de donde el sistema tendrá que obtener los resultados que necesita otro u otros procesos. Por último revela la posición en la cola del proceso en la que se tiene que depositar los resultados previamente adquiridos. Repitiendo esta secuencia en la información que se indica para los demás procesos dependientes.

Las siguientes filas en el archivo de configuración contienen información con la misma estructura que se explicó en la fila número 2, sólo cambian los nombres de los procesos y sus dependencias.
De igual manera, cada archivo de configuración tendrá que tener la misma estructura, ya que el funcionamiento del sistema depende de ello.

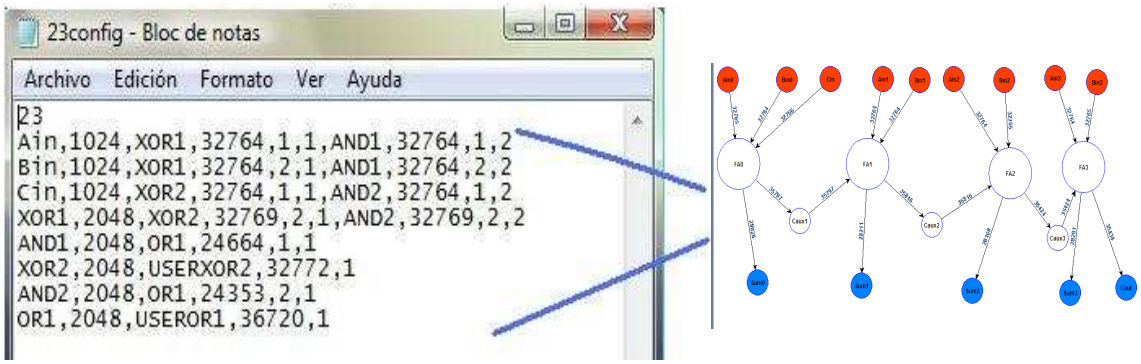

Figura 3.4. Estructura del archivo de configuración.

3.3.2. CREACIÓN DE LA PLATAFORMA ESTÁTICA DEL SISTEMA

Es aquí donde se definen y crean los elementos que configuran un ambiente estático para cada ejecución de una aplicaciones en forma distribuida, a la hora de lanzar la ejecución de una aplicación, la plataforma se adecua para atender las necesidades de la aplicación. La plataforma consta de la creación de una cola de resultados donde se almacenan los resultados que generen ciertos procesos, para que puedan ser observados por el usuario. Además de esto, es aquí donde se obtiene el PID de la aplicación, con el cual se obtienen los eventos que se generen en la bitácora durante la ejecución de ésta, logrando con ello la presentación al usuario de todos los eventos generados para el PID de la aplicación que se ejecutó. Por último, es en este punto donde se crea un contenedor de procesos, el cual contiene toda la información obtenida del archivo de configuración más los procesos que van a ser ejecutados, y es enviado al maestro para que se encargue de distribuir el trabajo entre sus trabajadores.

3.3.3. CONTENEDOR DE PROCESOS

Es un elemento creado por el sistema para encapsular información y datos, los cuales pertenecen a una aplicación que va a ser ejecutada por el sistema. Este contenedor es llenado con la información que se obtiene del archivo de configuración y con los procesos, a continuación se muestra la estructura de este contenedor (Véase Figura 3.5).

Figura 3.5. Contenedor de procesos.

El contenedor de procesos es estructuralmente un tipo de dato perteneciente al lenguaje de programación C# que representa una colección (ArrayList), contiene *n* elementos, cada uno de ellos es utilizado dentro la planificación, distribución de procesos y el balanceo de las cargas de trabajo. A continuación se describe cada elemento del contenedor.

- **PID:** Ayuda a diferenciar las aplicaciones que se estén ejecutando, a pesar de que ciertos procesos pertenecientes a diferentes aplicaciones tengan los mismos nombres.
- **Nombre del proceso:** Permite identificar a los procesos que serán ejecutados.

- **Proceso:** Forma parte de una aplicación, se encarga de crear el sistema de colas, realiza lecturas y escrituras, además de que procesa los datos obtenidos en ellas.
- **Peso del proceso:** Indica que procesos generaran más procesamiento que otros, por lo tanto sus requerimientos.
- **Dependencias:** Este campo contiene una serie de información que se encargará de informar al trabajador la dirección IP a donde tiene que transportar los resultados que el proceso en ejecución genere, dicha información está organizada de la siguiente manera:
    - **Campo 1.-**Nombre de la cola de almacenamiento a la que se agregarán los resultados generados por algún proceso en ejecución.
    - **Campo 2.-** Número promedio de conexiones realizadas entre los procesos dependientes.
    - **Campo 3.-** Número de fila de la cual tendrá que escribir el resultado que generó el proceso en cuestión.
    - **Campo 4.-** Número de fila de la cual leerá el resultado generado por el proceso anteriormente mencionado para enviárselo a otro proceso que lo ocupe como entrada.

En este contenedor se almacenan todos los procesos que conforman la aplicación que se va a ejecutar.

3.4.MAESTRO

Este es el encargado de obtener recursos de todas aquellas computadoras que formen parte de la red, logrando utilizar la capacidad de procesamiento que tiene cada una de ellas en su beneficio para realizar la ejecución de procesos que pertenecen a aplicaciones intensivas. Es a partir de este elemento donde se puede observar que el sistema procede a interactuar en forma de Computación en Malla.

El maestro consta de tres componentes los cuales interactúan entre sí para realizar el balanceo de las cargas de trabajo y la distribución de procesos. A continuación se describe cada componente.

3.4.1. COMPONENTE ASOCIACIÓN

Se encarga de obtener una lista de los trabajadores disponibles, además de sus recursos, entre los que se encuentran: 1) Espacio libre en la unidad C. 2) Tamaño, velocidad y cantidad de RAM libre. 3) Número de procesadores lógicos y velocidad de cada uno de ellos.

Para lo cual el componente abre el archivo ubicado en la ruta C:/INOUTQUEUES/ConfiguracionTrabajadores/ListaTrabajadores.txt" y obtiene las direcciones IP de todos los trabajadores que se encuentran registrados en ese archivo.

Posteriormente verifica la disponibilidad de cada uno de ellos, para lo cual se encarga de intentar establecer la comunicación con cada uno de los trabajadores. En caso de no poder establecerla con algún trabajador, el componente procede a descártalo; para todos aquellos trabajadores con los cuales se haya logrado entablar la comunicación, el componente obtiene sus recursos y los agrega a una nueva lista de trabajadores disponibles.

3.4.2. COMPONENTE GRAFO DEL FLUJO DE DATOS

Tiene como objetivo crear un grafo donde se encuentran las direcciones IP de todos los trabajadores disponibles, nombres y el número total de procesos asignados a cada trabajador.

El componente clasifica a cada uno de los trabajadores en: 1) Trabajadores con alto poder de cómputo. 2) Trabajadores con bajo poder de cómputo. Esta clasificación se da gracias a la obtención de sus recursos mediante el componente asociación la cual se realiza de la siguiente manera: la multiplicación del número de procesadores lógicos por la velocidad de cada uno de ellos más la velocidad de la RAM, esto para cada trabajador. Después de realizar dicha operación, verifica que el número obtenido como resultado sea mayor a 900; si es este el caso, se clasifica al trabajador con alto poder de cómputo, en caso contrario con bajo poder de cómputo.

Una vez hecho lo anterior el componente procede a realizar la clasificación de los procesos, para lo cual verifica el peso de cada uno para obtener la media aritmética de todos los pesos, de esta manera aquel proceso que tenga un peso menor o igual a la media, se clasifica como proceso ligero, en caso contrario se clasifica como proceso pesado.

Por último asigna los procesos a los trabajadores teniendo en cuenta las siguientes situaciones: 1) Siempre asignará un proceso a aquel trabajador que tenga menos procesos asignados para ejecución. 2) A los trabajadores con bajo poder de cómputo sólo se les podrán asignar procesos ligeros, verificando siempre la situación 1. 3) A los trabajadores con alto poder de cómputo se les podrán asignar tanto procesos ligeros como pesados, verificando siempre la situación 1.

3.4.3. CONTENEDOR UNITARIO DE PROCESO

El contenedor unitario de proceso al igual que el anterior, es un elemento creado por el sistema para encapsular información y datos pertenecientes a una aplicación que va a ser ejecutada por el sistema; teniendo como diferencia al anterior, que este contenedor es enviado a los trabajadores para que realicen la ejecución del proceso que se encuentra dentro, además sólo consta de cinco campos, donde se incluye información de la ubicación a la que tiene que transportar los resultados que el proceso genere, (Véase Figura 3.6).

Figura 3.6. Estructura del contenedor unitario de proceso.

El contenedor unitario de proceso es estructuralmente un tipo de dato perteneciente al lenguaje de programación C# que representa una colección (ArrayList), contiene 5 elementos, cada uno de ellos es utilizado dentro del proceso de ejecución y almacenamiento de resultados. A continuación se describe cada elemento del contenedor unitario de proceso.

- **PID:** Ayuda a diferenciar las aplicaciones que se estén ejecutando, a pesar de que ciertos procesos pertenecientes a diferentes aplicaciones tengan los mismos nombres.
- **Nombre del proceso:** Permite identificar a los procesos que serán ejecutados.
- **Proceso:** Forma parte de una aplicación, se encarga de crear el sistema de colas, realiza lecturas y escrituras, además de que procesa los datos obtenidos en ellas.
- **Dependencias:** Este campo contiene una serie de información que se encargará de informar al trabajador la dirección IP a donde tiene que transportar los resultados que el proceso en ejecución genere, dicha información está organizada de la siguiente manera:

- **Campo 1.-**Nombre de la cola de almacenamiento a la que se agregarán los resultados generados por algún proceso en ejecución.
- **Campo 2.-** Dirección IP donde se encuentra el proceso dependiente.
- **Campo 3.-** Número de fila de la cola donde tendrá que escribir el resultado que genero el proceso en cuestión.
- **Campo 4.-** Número de fila de la cola donde leerá el resultado generado por el proceso anteriormente mencionado para enviárselo a otro proceso que lo ocupe como entrada.

- **IP del maestro:** La cual se utiliza para se envíen los eventos que se deben almacenar en la bitácora

Cabe mencionar que es la misma estructura que se sigue para todos los contenedores unitarios de proceso.

3.4.4. COMPONENTE PLANIFICADOR

Es el encargado de realizar la distribución de procesos y balanceo de las cargas de trabajo entre los trabajadores disponibles pertenecientes a la Computación en Malla, para lo cual crea un contenedor unitario de proceso por cada proceso existente.

Es aquí donde realiza una modificación al campo 2 de las dependencias, sustituyendo su contenido por la dirección IP, ya sea del trabajador o del usuario dependiendo de la cola donde se tienen que depositar los resultados; si el nombre de la cola contiene la palabra USER al inicio, se identifica que los resultados se enviarán al usuario, en caso contrario se enviarán a algún trabajador.

Para finalizar, el componente realiza una conexión con cada uno de los trabajadores disponibles para que se realice la entrega del contendor unitario de proceso.

3.5. TRABAJADOR

Es el encargado de realizar la ejecución de los procesos que le asigne el maestro, además de ello se encarga de realizar el transporte de los resultados de cada proceso que tenga en ejecución a la cola de resultados de entrada de otros procesos o a la cola de resultados donde se encuentra la interfaz de usuario dependiendo de la dirección IP que se le haya indicado en el campo 2, de las dependencias

que se encuentran dentro del contenedor unitario de proceso. Siendo esta parte del sistema la que crea el modelo para el paso de mensajes entre procesos utilizando la tecnología punto a punto.

3.5.1. SISTEMA DE COLAS

Este sistema es creado por cada proceso que pertenece a una aplicación que va a ser ejecutada en el sistema, el diseño de éstas es:

Cada proceso tiene que crear una cola tanto de entrada como de salida por cada otro proceso que necesite sus resultados, esto se puede observar de mejor manera por medio de la Figura 3.7, la cual presenta el grafo de comunicaciones que existen entre procesos para una cierta aplicación.

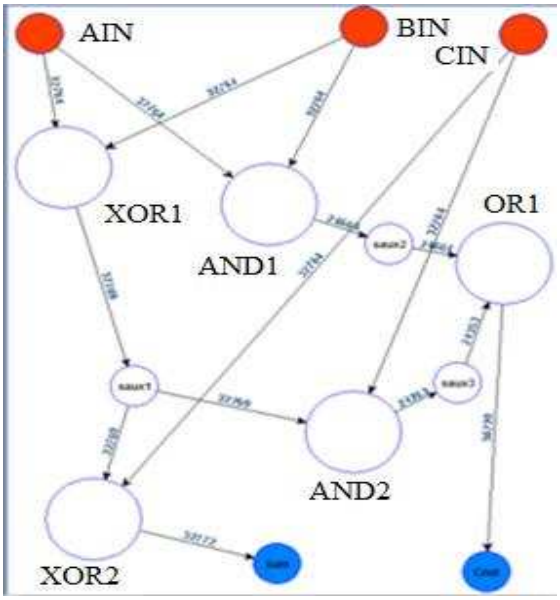

Figura 3.7. Grafo que describe la comunicación entre procesos de una aplicación.

Utilizando como un ejemplo para describir de mejor manera la creación de las colas se procede a ocupar la Figura 3.7. El proceso llamado XOR1 tiene que crear una dos colas de entrada para recibir los resultados que le envíen los procesos AIN y BIN y dos colas de salida para los procesos XOR2 y AND2, ya que en éstas escribe los resultados que tienen que recibir estos procesos. De igual manera, el proceso llamado AND1 debe crear dos colas de entrada para recibir los resultados que envíen los procesos AIN y BIN, y en este caso sólo debe crear una cola de salida para el proceso OR1, ya que en éstas escribe los resultados que tiene que recibir este proceso. Es importante denotar

que se sigue el concepto de creación de colas para cada uno de los demás procesos existentes en el grafo; de igual manera para cualquier grafo de comunicaciones de otra aplicación.

3.5.2. COMPONENTE EJECUTOR

Tiene como objetivo almacenar y lanzar la ejecución de los procesos que recibe, para lo cual el componente recibe todos los contenedores unitarios de proceso enviados por el maestro, posteriormente procede a realizar el almacenamiento de los procesos en un directorio de trabajo, así que se posiciona en la ruta "C:/INOUTQUEUES/" donde reconstruye al proceso, una vez terminado lo anterior realiza el envío del evento de almacenamiento del proceso al maestro, para que sea agregado en la bitácora.

En seguida lanza la ejecución del proceso, de igual manera como si se hiciera por medio de la consola DOS. A la hora de lanzar la ejecución del proceso se le pasa como parámetro el PID, ya que cada proceso lo necesita para poder crear los nombres de las colas de entrada y salida para los resultados; después de ello, realiza el envío del evento de ejecución del proceso al maestro, para que sea agregado en la bitácora.

También se encarga de transportar los resultados de los procesos que están en ejecución, ya sea que los envíe a otros trabajadores o a la interfaz de usuario dependiendo de lo indicado en el campo de dependencias. Para realizar esta tarea primeramente intenta abrir la cola de resultados de salida generada por el proceso en ejecución, en caso de que no pueda abrirla, ya sea porque aún no está creada o porque se encuentra abierta por otro proceso; el componente espera medio segundo y vuelve a intentar abrirla, realizando este ciclo hasta que lo logre, ya abierta obtiene de cada línea los resultados y los almacena, después borra dichos resultados de la cola; en caso de que alguna línea no contenga ningún resultado, procede nuevamente a esperar medio segundo y después vuelve a realizar las tareas anteriormente mencionadas.

Una vez obtenidos los resultados el componente identifica si el nombre de la cola del proceso dependiente, contiene la palabra USER; en caso afirmativo envía a la interfaz de usuario la siguiente información: 1) Nombre de la cola donde tiene que agregar los resultados. 2) Número de línea donde tiene que escribir los resultados. 3) Los resultados. 4) El PID. En caso de que no

contenga la palabra USER el nombre de la cola del proceso dependiente, se envían los datos mencionados al trabajador que está ejecutando el proceso dependiente.

Ya que el trabajador o la interfaz de usuario recibieron los datos junto con la información, el componente ejecutor local intenta abrir la cola de resultados del proceso dependiente. En caso de que no pueda abrir la cola, ya sea porque aún no está creada o porque se encuentra abierta por otro proceso; el componente espera medio segundo y vuelve a intentarlo, realizando este ciclo hasta que logre abrirla, posteriormente ya abierta, agrega los resultados a la cola en la línea indicada.

Capítulo 4

Pruebas y Resultados

Se proporciona una perspectiva general de los resultados obtenidos en cuanto al funcionamiento del sistema, además se describen las pruebas realizadas para medir el rendimiento, así como las características de cada equipo de computo utilizado durante el periodo de pruebas.

El contenido de los temas presentados en este capítulo incluye:

- Presentación de resultados.
- Presentación de pruebas que evalúan el rendimiento del sistema.
- Descripción de la generación de la prueba (sumador de 1 bit con acarreo).
- Descripción de la generación de la prueba (localizador de rutas).

4.1 PRESENTACIÓN DE RESULTADOS

Con motivo de poder especificar a fondo los resultados que se obtuvieron en cuanto al funcionamiento del sistema, en el presente capitulo mostramos nuestros resultados utilizando dos benchamarks (Bancos de pruebas) que nos permiten evaluar y caracterizar el sistema desarrollado. Uno de los bancos de pruebas genera la simulación de un circuito digital (sumador de 1 bit con acarreo), este fue particionado en 8 procesos los cuales interactúan entre sí; para esta prueba se ocuparon 3 computadoras participantes con las siguientes características. (Véase Tabla 4.1). Dichas computadoras están conectadas a una LAN. El otro banco de pruebas realiza la simulación que permite localizar la Ruta más Corta entre dos puntos de una cuadricula que emula un ciudad.

Tabla 4.1. Características de las computadoras pertenecientes a la Malla.

IP	CARACTERISTICAS	DESEMPEÑO DE TRABAJO COMO:
192.168.1.65	-Velocidad de Procesador: **2 GHz.** -Tamaño de la RAM: **2.00 GB.** -Velocidad de la RAM: **667MHz.** -Tamaño del Disco Duro: **100 GB.** -Sistema Operativo: **Windows Vista Home Premium.**	**Usuario y Maestro**
192.168.1.68	-Velocidad de Procesador: **400 MHz.** -Tamaño de la RAM: **200 MB.** -Velocidad de la RAM: **70MHz.** -Tamaño del Disco Duro: **40 GB.** -Sistema Operativo: **Windows 2000 Profesional SP 4.**	**Trabajador**
192.168.1.73	-Velocidad de Procesador: **3 GHz** -Tamaño de la RAM: **256 MB.** -Velocidad de la RAM: **400 MHz.** -Tamaño del Disco Duro: **40 GB.** -Sistema Operativo: **Windows XP SP 2.**	**Trabajador**

4.1.1 CARGA DEL ARCHIVO DE CONFIGURACIÓN DE LA APLICACIÓN Y CREACIÓN DEL CONTENEDOR DE PROCESOS

Como primer paso el sistema se encarga de procesar el archivo de configuración de la simulación, al cual se le asignó el PID (Process Identifier, Identificador de Proceso) número 23.

Una vez que se ha obtenido la información del archivo de configuración, el sistema, se encarga de crear un contenedor de procesos donde almacena la información obtenida de dicho archivo más los procesos que conforman a la aplicación.

NOTA: Cuando el sistema detecta que el nombre de algún proceso tiene al inicio la palabra USER, procede a crear una cola donde se almacenarán los resultados que tal proceso genere, también agrega información adicional, al contenedor de procesos para que cuando ejecuten ese proceso, los resultados que genere se almacenen en la cola especificada en la parte de dependencias del contenedor de procesos. Esta tarea se realiza en forma dinámica para cada proceso que vaya a generar resultados que el usuario deba observar.

Una vez realizada las tareas anteriormente mencionadas, el sistema procede a mostrar el contenido del contenedor de procesos a través de la interfaz de usuario. En la Figura 4.1 se muestra una comparación del archivo de configuración de la aplicación y el contenido del contenedor de procesos, donde se pueden observar las modificaciones (encerradas en un círculo amarillo), que el sistema realizó a la hora de crear dicho contenedor.

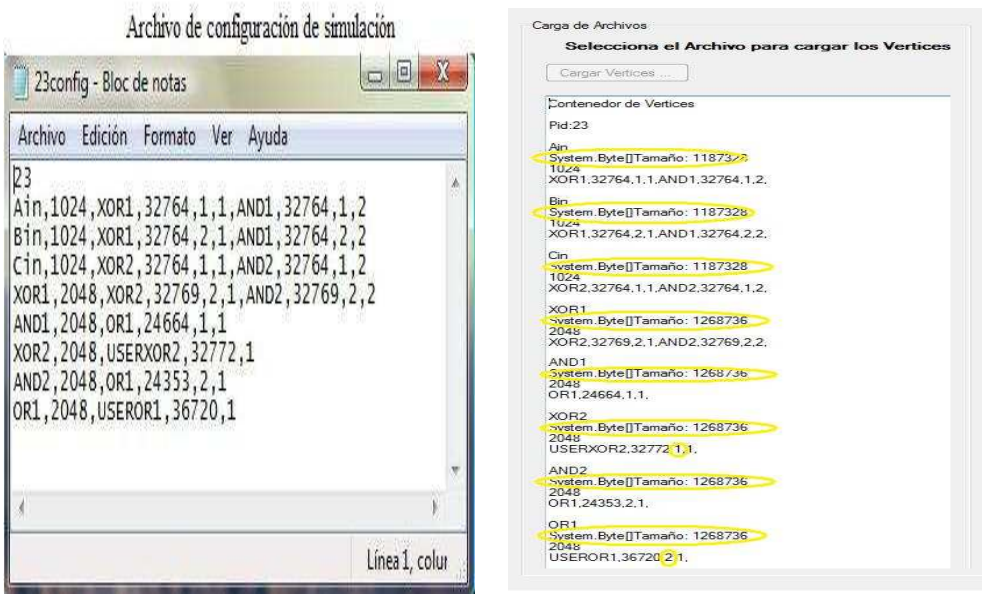

Figura 4.1. Comparación entre el contenido del archivo de configuración y el contenido contenedor de vértices.

4.1.2 CREACIÓN DE COLA DE RESULTADOS

La creación de la cola de resultados se realiza mientras se está creando el archivo de configuración, ya que es en este punto donde se identifican aquellos procesos que van a entregar resultados para el Usuario (En este caso existen dos que entregan resultados al Usuario: *XOR2* y *OR1*). Tal cola de resultados contiene un nombre compuesto por tres elementos: el *PID* de la simulación más la palabra *Resultados* más el indicativo de que la cola es de entrada (*IN)*. Posteriormente a esto, escribe en la cola el nombre del proceso, ya sin la palabra USER; esto se realiza para que el usuario sepa qué resultados pertenecen a qué procesos, la Figura 4.2 muestra la cola de resultados creada por el sistema.

Figura 4.2. Cola de resultados perteneciente a la simulación del sumador de 1 bit con acarreo, a la cual se le asigno el PID 23.

4.1.3 ENVIÓ DEL CONTENEDOR DE PROCESOS AL MAESTRO

Ya que el contenedor de procesos ha sido creado, se procede a realizar el envió de este al maestro, por tanto, el sistema verifica si puede establecer comunicación con el maestro, lo que genera dos situaciones:

- **La comunicación no se logra establecer:** Esto generaría un error el cual se mostraría al usuario mediante la interfaz de usuario.
- **La comunicación se establece correctamente:** Lo que permite realizar el envió del contenedor de procesos.

4.1.4 PLANIFICACIÓN, DISTRIBUCIÓN DE PROCESOS Y BALANCEO DE LAS CARGAS DE TRABAJO

Estas acciones son realizadas por los componentes pertenecientes al maestro. Primeramente el sistema se encarga de utilizar el método de distribución de las cargas de trabajo (basado en los requerimientos de cada proceso y los recursos disponibles en cada unidad de ejecución); lo que el sistema realiza en este momento es una clasificación de procesos, lo cuales conforman a la aplicación basándose precisamente en sus pesos (requerimientos de cada proceso).

El sistema al obtener dichos pesos se encarga de realizar la ordenación en forma ascendente de ellos, para que posteriormente obtenga de todos esos valores la media aritmética, esta se ocupa para realizar la siguiente clasificación:

- Procesos que tengan un peso igual o menor a la media obtenida, se clasifican como procesos LIGEROS.
- Procesos que tengan un peso mayor a la media obtenida, se clasifican como procesos PESADOS.

Esta clasificación es realizada cada vez que una aplicación va ha ser ejecutada por el sistema. Para este caso arroja como resultado de la media aritmética obtenida el valor de 2048 como lo muestra la Figura 4.3. Es importante denotar que los procesos tienen los siguientes pesos: **Proceso 1:** 1024, **Proceso 2:** 1024, **Proceso 3:** 1024, **Proceso 4:** 2048, **Proceso 5:** 2048, **Proceso 6:** 2048, **Proceso 7:** 2048, **Proceso 8:** 2048.

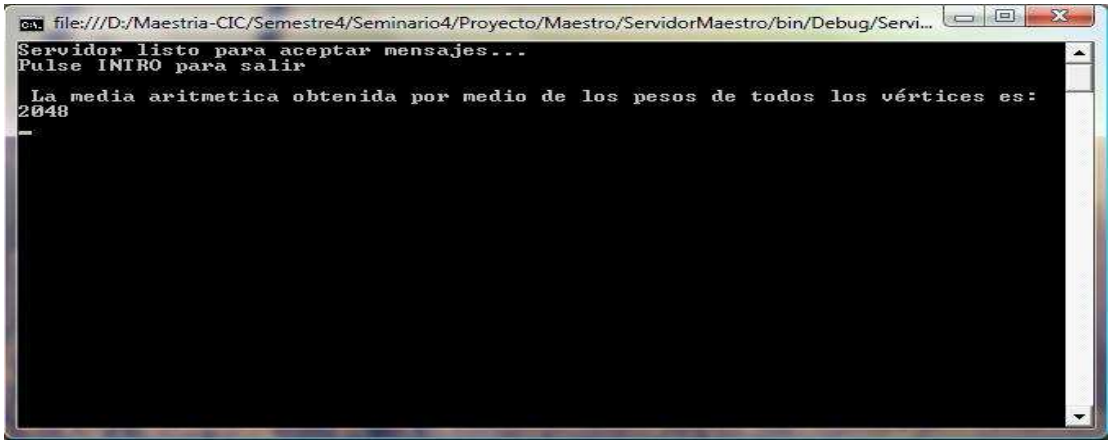

Figura 4.3. Media aritmética obtenida por los pesos de 8 procesos pertenecientes a la aplicación (sumador de 1 bit con acarreo).

4.1.5 OBTENCIÓN DE LA LISTA DE TRABAJADORES DISPONIBLES

Primero el sistema se encarga de obtener por medio de un archivo de texto las direcciones IP de todos los trabajadores pertenecientes a la Malla, enseguida de ello intenta establecer la comunicación con cada uno de los trabajadores. Para aquellos con los que logre establecer la comunicación, los agrega a una lista junto con sus recursos como lo muestra la Figura 4.4; en el caso de aquellos trabajadores con los cuales no logró establecer comunicación, el sistema procede a descartarlos.

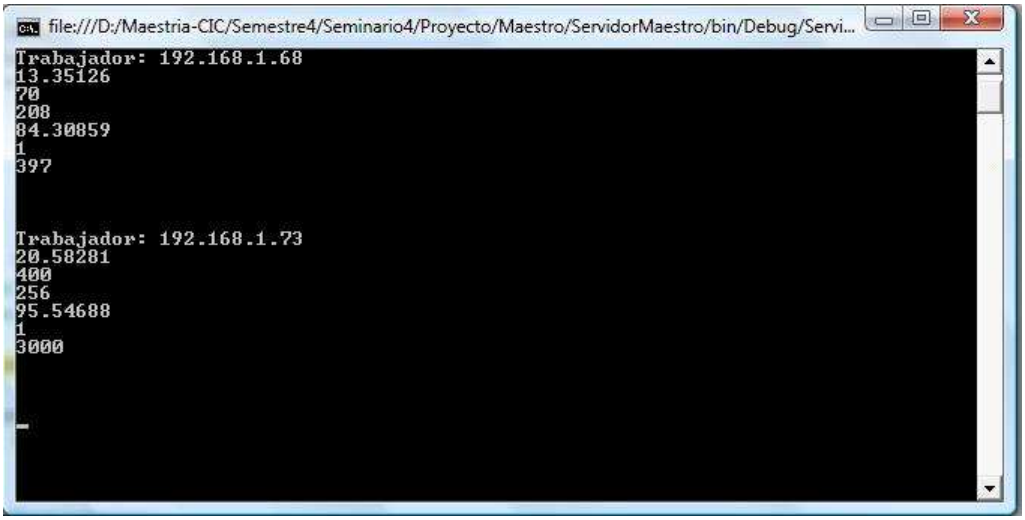

Figura 4.4. Características de los trabajadores disponibles pertenecientes a la Malla.

La Figura 4.4 permite observar las características que tiene cada trabajador, mostrándolas en forma de lista: espacio libre en la unidad local C, velocidad de la RAM, Tamaño de la RAM, tamaño del espacio libre en la RAM, número de procesadores, velocidad del procesador. Estos recursos son utilizados por el sistema para clasificar a los trabajadores.

4.1.6 CREACIÓN DEL GRAFO DE FLUJO DE DATOS

El sistema se encarga de clasificar a cada trabajador de la siguiente manera:

- **Trabajadores con bajo poder de computo:** Si la multiplicación del número de procesadores por la velocidad de cada uno de ellos, más la velocidad de la memoria, da como resultado un número menor a 900.

- **Trabajadores con alto poder de computo:** Si la multiplicación del número de procesadores por la velocidad de cada uno de ellos, más la velocidad de la memoria, da como resultado un número mayor a 900.

En la Figura 4.5 se puede observar que el sistema ha clasificado al trabajador 192.168.1.68 con bajo poder de cómputo debido a que la operación aritmética con sus recursos, mencionada anteriormente, no superó la cifra de 900.

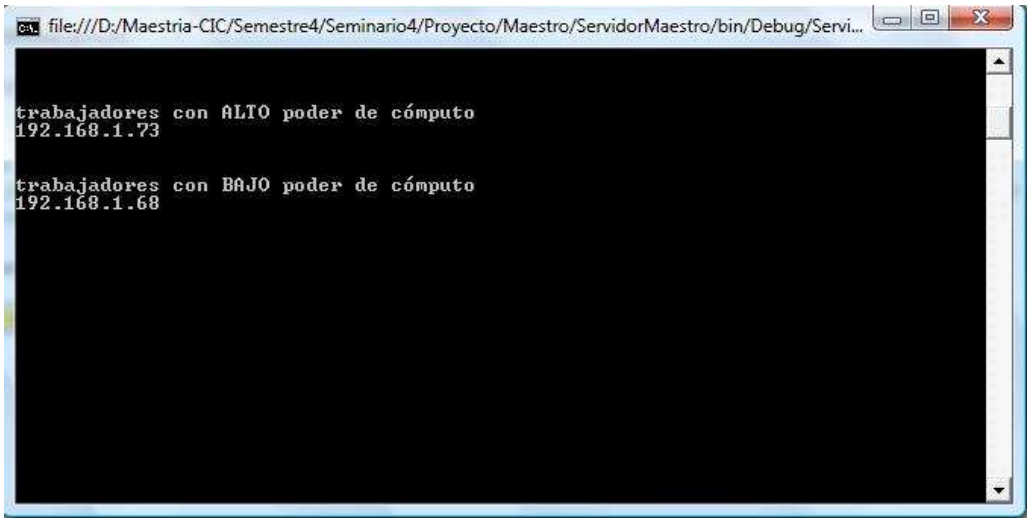

Figura 4.5. Clasificación de trabajadores.

Posteriormente asigna los procesos a los trabajadores. Dando como resultado lo que muestra la Figura 4.6, permite observar que al trabajador *192.168.1.68* el sistema sólo le asignó 2 procesos, esto se da debido a que el sistema identificó que los pesos de los procesos *AIN* y *CIN* están por debajo de la media aritmética obtenida anteriormente, por lo cual los clasificó como ligeros, además de ello, este trabajador fue clasificado con bajo poder de computo, lo que quiere decir que sólo se le van a asignar procesos ligeros. Al trabajador *192.168.1.73* se le asignaron 6 procesos, ya que este fue clasificado con alto poder de cómputo, lo que quiere decir que puede ejecutar tanto procesos ligeros como pesados.

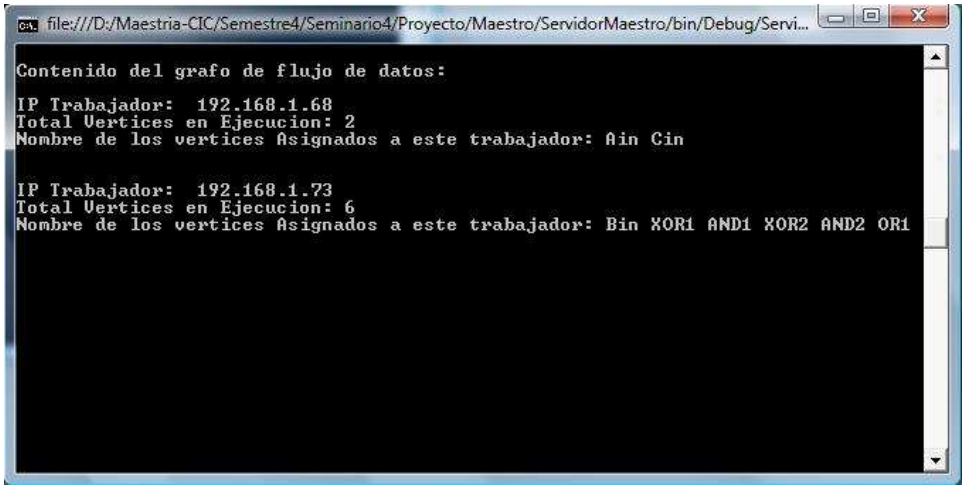

Figura 4.6. Asignación de procesos a trabajadores.

4.1.7 ENVIÓ DE LOS PROCESOS A LOS TRABAJADORES

Es ahora cuando el sistema procede a crear un nuevo contenedor unitario de proceso, por cada proceso a distribuir.

Enseguida, intenta establecer la comunicación con cada trabajador para realizar el envío de los contenedores unitarios de proceso, (Véase Figura 4.7).

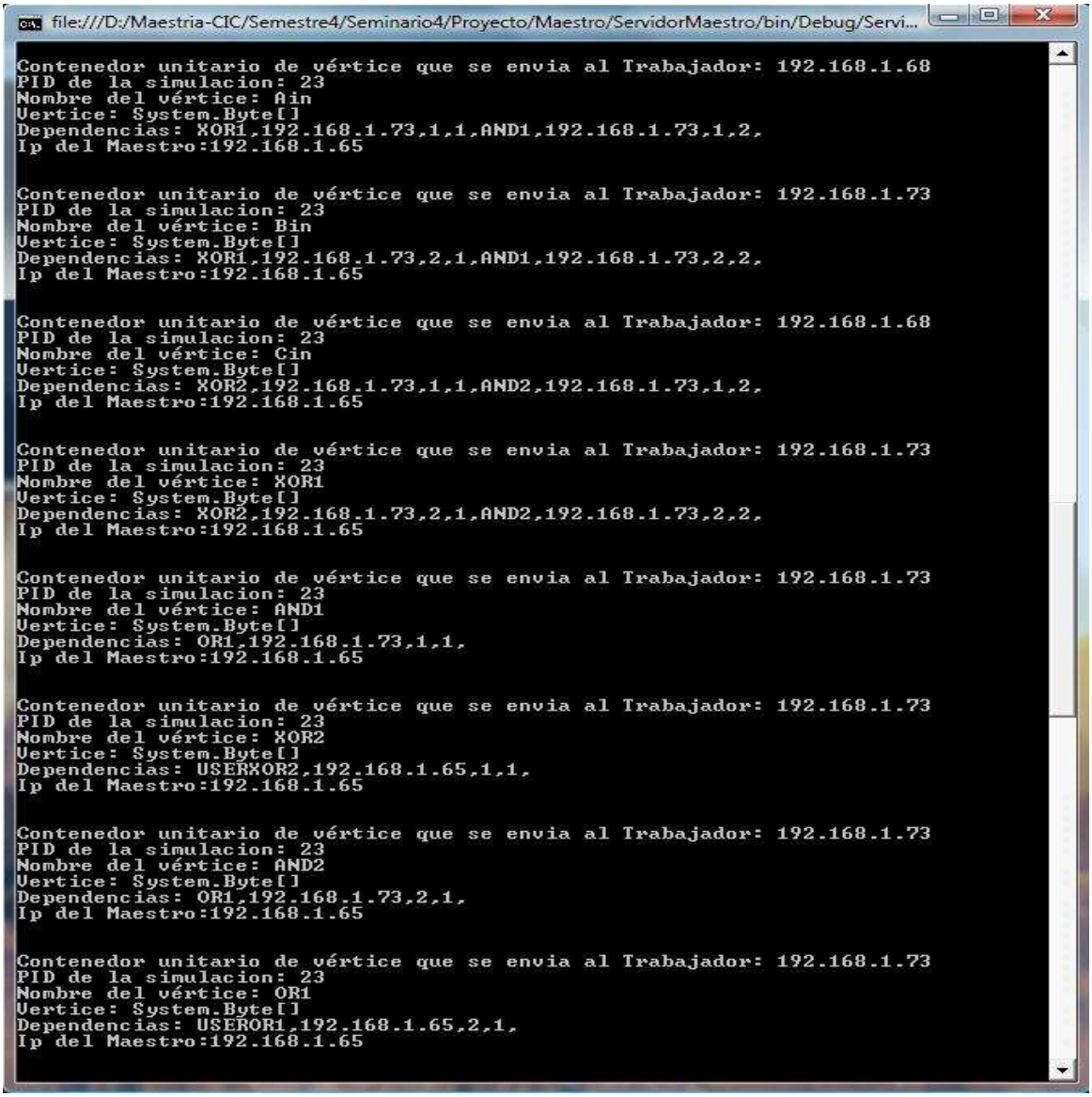

Figura 4.7. Creación y envió de los contenedores unitarios de proceso a los trabajadores.

Dentro de cada contenedor se encuentra la información y datos necesarios para que cada trabajador ejecute los procesos que se le asignaron y además de ello envíe los resultados generados por estos al trabajador que tenga en ejecución a otros procesos que necesiten estos resultados. Lo anterior se logra gracias a que las dependencias se actualizaron con la dirección IP de donde están el o los procesos dependientes, de esta manera se realiza la comunicación en forma punto a punto entre trabajadores e interfaz de usuario para la entrega de resultados.

4.1.8 EJECUCIÓN DE LOS PROCESOS

Estas acciones son realizadas por los componentes pertenecientes a los trabajadores. Una vez que un trabajador ha recibido el contenedor unitario de proceso, el sistema extrae los elementos de este contendor. Enseguida reconstruye el proceso y lo almacena en el directorio "C:/INOUTQUEUES/", terminada dicha tarea envía el evento de almacenamiento de proceso al maestro para que este lo almacene en la bitácora.

Dando como resultado que en el trabajador *192.168.1.73* se almacenaron 6 procesos los cuales crearon sus colas tanto de entrada como de salida. Por tanto en el trabajador *192.168.1.68* se almacenaron los 2 procesos restantes (Véase Figura 4.8 y 4.9).

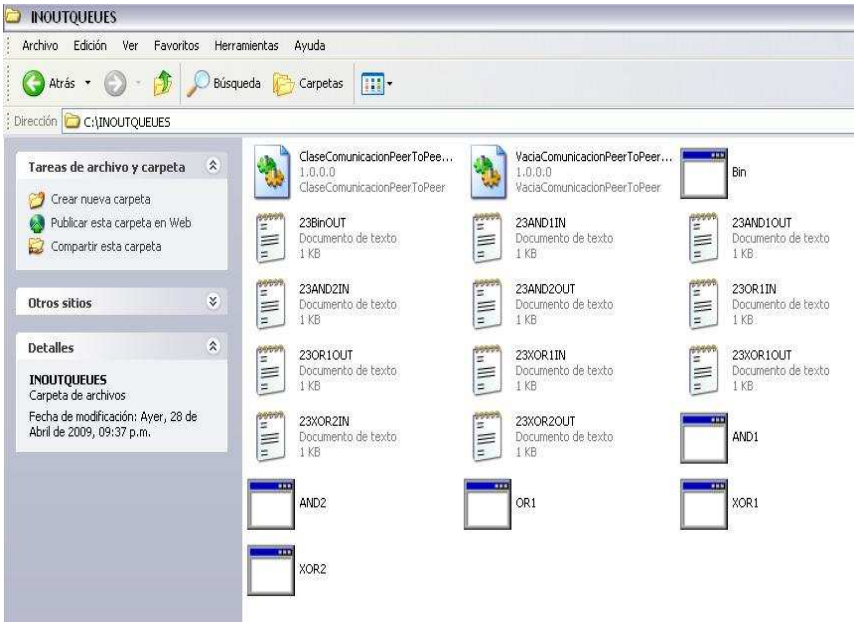

Figura 4.8. Almacenamiento de los procesos en el trabajador *192.168.1.73.*

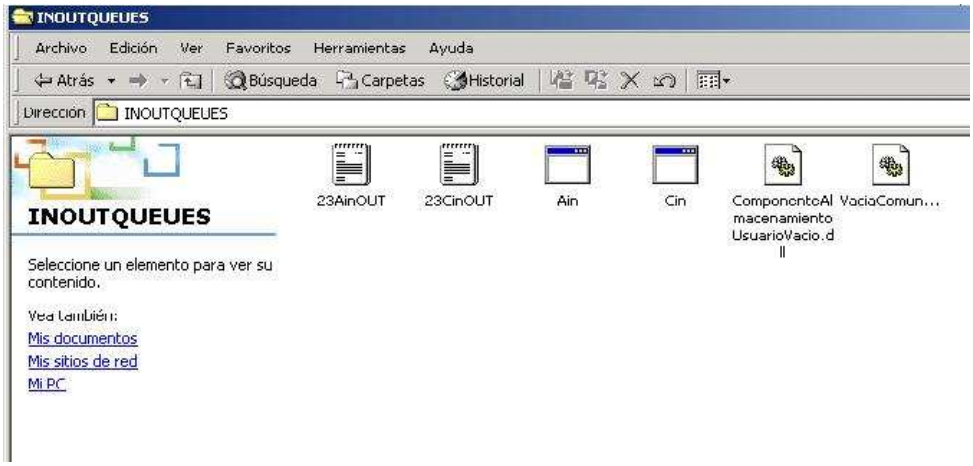

Figura 4.9. Almacenamiento de los procesos en el trabajador *192.168.1.68*.

Ahora el sistema procede a lanzar la ejecución de los procesos, a los cuales les pasa como parámetro el PID de la simulación al que pertenecen. Además de ello envía el evento de ejecución al maestro para que sea almacenado en la bitácora.

En la Figura 4.10 se puede observar que los procesos comienzan a tratar de leer o escribir en sus respectivas colas, a las cuales el sistema debe hacer llegar los datos que cada proceso tiene que computar, un momento después de que el sistema lanzo su ejecución.

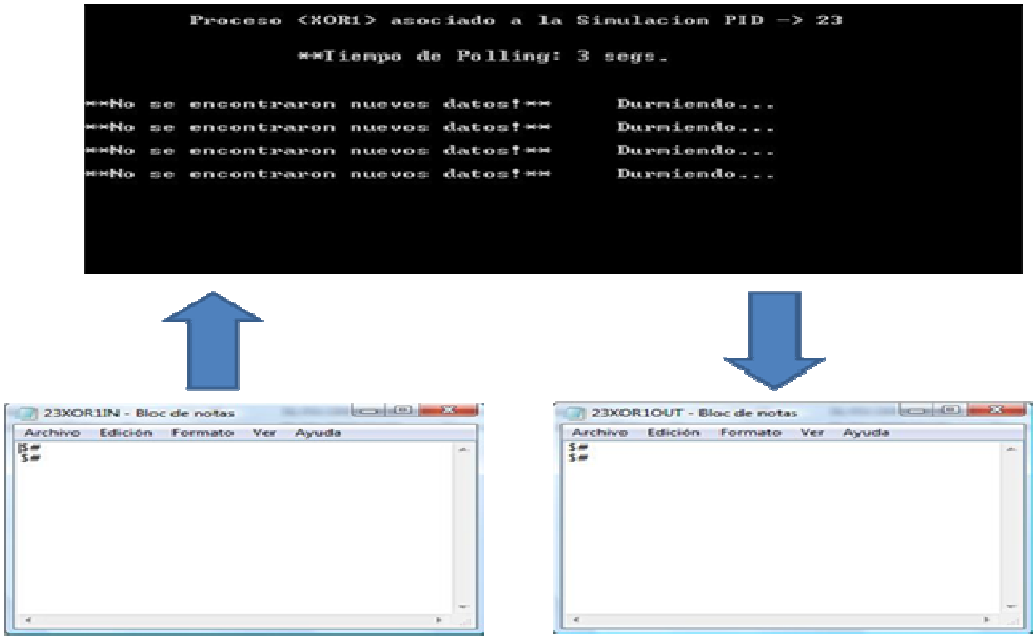

Figura 4.10. Lectura y escritura en las colas de cada proceso.

Es en este momento cuando el sistema envía el evento de ejecución del proceso al maestro.

4.1.9 SOLUCIONANDO LA DEPENDENCIA DE DATOS.

El sistema obtiene los resultados de la cola de salida de cada proceso y los envía a la dirección IP que se le indicó en las dependencias que se encuentran dentro del contenedor unitario de proceso, para que sean almacenados en la cola de entrada del proceso que utiliza dichos resultados (Véase Figura 4.11). Es en este punto donde el transporte de resultados, ya sea entre trabajadores o trabajadores – interfaz de usuario, se realiza en forma punto a punto.

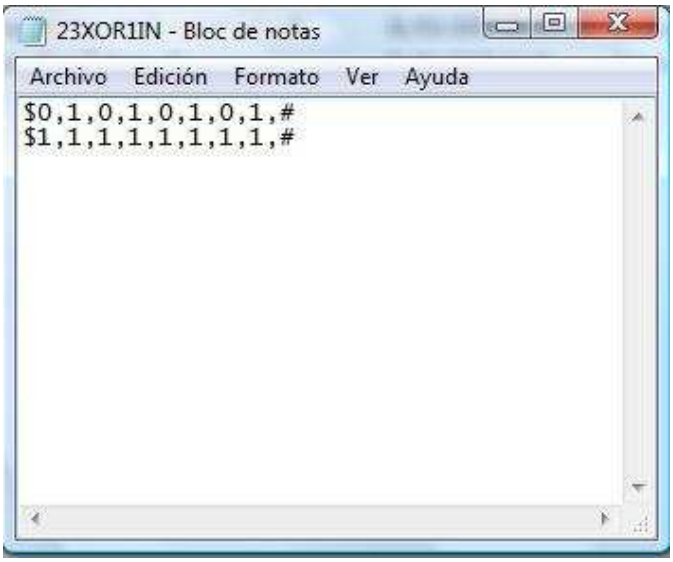

Figura 4.11. Agregado de los resultados en la cola de entrada de un proceso.

4.1.10 PRESENTACIÓN DE RESULTADOS AL USUARIO

En este punto el sistema se encarga copiar el contenido de la cola de resultados para posteriormente mostrarlo a través de la interfaz de usuario, logrando con ello que el usuario pueda observar los resultados que se vayan generando (Véase Figura 4.12).

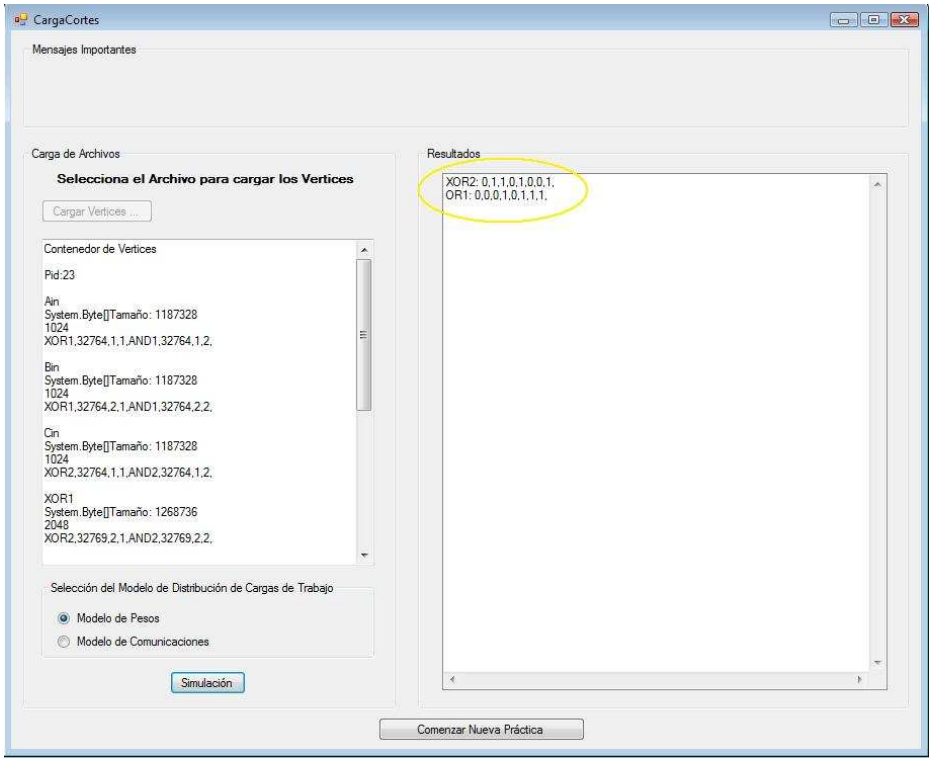

Figura 4.12. Presentación de resultados al usuario.

4.1.11 PRESENTACIÓN DE LA BITÁCORA AL USUARIO

Como primer paso el sistema obtiene el PID de la aplicación ya que gracias a este se pueden recuperar todos los eventos generados durante la simulación. Como resultado, se obtienen todos los eventos registrados en la bitácora para el PID 23, el cual conformó la aplicación del sumador de 1 bit con acarreo. (Véase Figura 4.13).

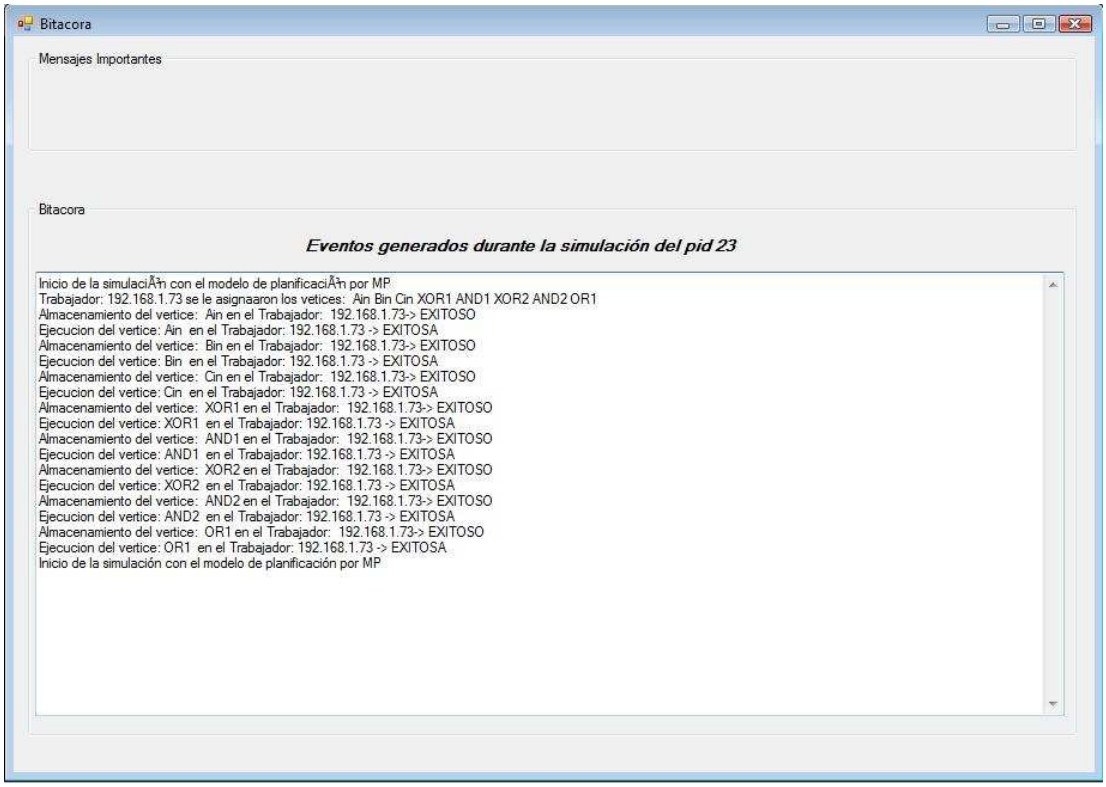

Figura 4.13. Presentación de la bitácora al usuario.

4.2 Presentación de Pruebas que evalúan el Rendimiento del Sistema

4.2.1 Descripción del Desarrollo de la Aplicación (Sumador de 1 Bit con Acarreo)

Descripción de la Arquitectura

El flujo de esta metodología puede ser descrita en 4 bloques principales:

- Código del diseño a ser verificado en lenguaje SystemC.
- Pre-simulación para monitoreo y construcción de estadística.
- Construcción del grafo que describe al sistema.
- Primer algoritmo de particionamiento por módulo.

En seguida se describe brevemente las funciones de cada bloque, considerando sus entradas y sus salidas para poder construir al final un mapa que permita figurar el esquema global de trabajo.

CÓDIGO EN SYSTEMC

Los lenguajes de descripción de hardware (HDL) se han utilizado tanto para diseños de arquitecturas en sistemas digitales como referencia en el análisis y validación de estos circuitos. Ofrecen una metodología de diseño jerárquica en las modalidades de *bottom-up* y *top-down*. En particular el lenguaje SystemC, permite programar en diferentes niveles de abstracción: nivel compuerta, nivel trasferencia de registros, nivel comportamiento, nivel transaccional, etc. Debido a que el nivel comportamiento representa un nivel aceptable (ni tan abstracto ni tan detallado) de programación (permite estructuras *case, if, for*), actualmente la mayoría de diseños de circuitos se realiza utilizando este nivel de abstracción.

En este proyecto se han aprovechado las ventajas del diseño jerárquico de los lenguajes de descripción de hardware en particular **SystemC**.

En esencia, SystemC es una biblioteca estandarizada por el IEEE portable en C++ y usada para modelar sistemas con comportamiento concurrente. SystemC suministra mecanismos valiosos para modelar hardware mientras se usa un ambiente compatible con el desarrollo de software. Este lenguaje también provee estructuras orientadas a modelar hardware que no están disponibles en los lenguajes de programación normales.

El motor de ejecución de SystemC permite la simulación de procesos concurrentes al igual que Verilog (Lenguaje de descripción de hardware), VHDL (Lenguaje de descripción de hardware) o cualquier otro lenguaje de descripción de hardware. Esto es posible gracias al uso de un modelo cooperativo multitarea, el cual corre en un kernel que orquesta el intercambio de procesos y mantiene los resultados y evaluaciones de dichos procesos alineados en tiempo.

Por otro lado, ya que C++ implementa la orientación a objetos (paradigma que de hecho fue creado para técnicas de diseño de Hardware) la abstracción de datos, propiedades, comportamientos y atributos de un modulo (clase) puede ser el principio para elaborar un diseño jerárquico.

A su vez, es posible describir el comportamiento de las entidades desde varios niveles de abstracción y puede ser explotado para obtener flujos de diseño y verificación más rápidos.

Una vez expuesto lo anterior, hay varias rutas por las cuales puede llegarse a tener el código de un sistema en SystemC listo para ser simulado: Traducido de algún otro lenguaje como VHDL o Verilog, o escrito directamente en SystemC. (Véase Figura 4.14).

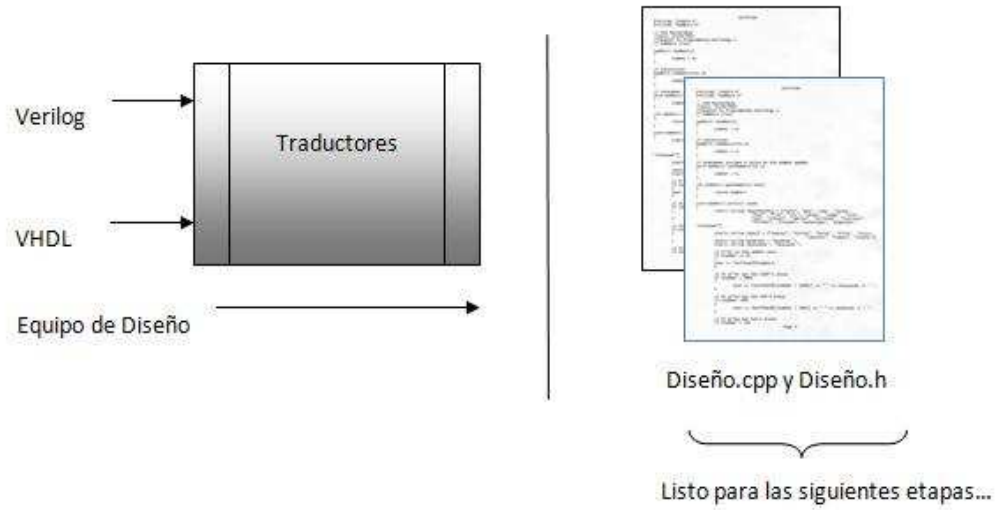

Figura 4.14. Conversión de código Verilog, VHDL a SystemC.

Hasta ahora se han probado varias alternativas de traducción y para Verilog, los traductores V2SC de Mazdak & Alborz Design Automation y Verilator de Veripool han pasado satisfactoriamente las traducciones de Verilog a SystemC.

Para VHDL, el traductor dentro del VSP Compiler resultó ser el mejor, aunque es una herramienta comercial (se corrió en una versión de prueba). Otras opciones son VH2SC o VHDL-to-SystemC-Converter de www-ti.informatik.uni son sin costo.

PRE-SIMULACIÓN

Se ha elaborado una biblioteca de monitores (SC_MONITOR.h) orientados a los tipos de datos usados en SystemC para diseño con el fin de descubrir aquellos bloques dentro de un diseño que presenten una suficiente cantidad de transacciones para poder ser particionados. La métrica para decidir cuál es la cantidad suficiente para particionar viene dada por un estudio estadístico de las transacciones que ocurren en el diseño basándose en un plan de verificación y puntos de cobertura.

El proceso de monitoreo pretende descubrir interfaces de mínima actividad que permita optimizar el particionamiento de los bloques donde se realizan altos niveles de transacciones. De este modo, la segunda etapa consiste en hacer un analizador del código para identificar todos los nodos (señales de entrada, de salida, de procesamiento, buffers, etc.) de los cuales se vale el sistema para procesar sus datos, (Véase Figura 4.15). Después de haberlos identificado, hay que colocar estas "puntas" de prueba sobre ellos de tal forma que midan en una pre-simulación la cantidad de transacciones que manejan todos y cada uno de esos nodos.

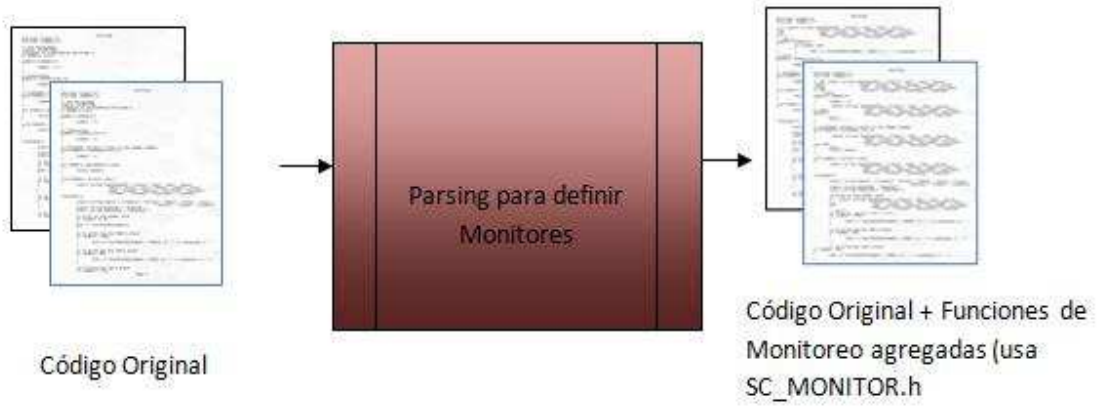

Figura 4.15. Analizando el código original.

[27] da una referencia confiable por la cual corriendo una simulación breve del diseño puede estimarse la carga computacional asociada con cada nodo. Durante esta pre simulación, un contador es mantenido para cada nodo del sistema, y cada vez que en alguno de ellos se detecta un cambio, el contador es incrementado.

Para las pruebas hechas, las medidas obtenidas en la evaluación funcional durante solo un 10% del total de la simulación resultaron consistentes con el comportamiento global de los nodos. Esto indica que nuestro algoritmo de particionamiento posterior puede descansar razonablemente en una pre simulación que permita predecir las cargas de trabajo que tendrá cada nodo (Véase Figura 4.16).

Al final, la pre simulación arroja, un vector en un archivo de texto y con ello los nombres de los nodos y la cantidad de transacciones hechas por cada uno de ellos.

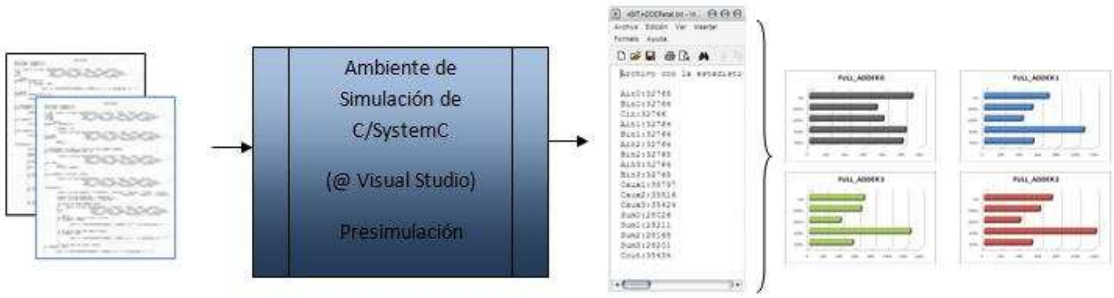

Figura 4.16. Pre simulación para la predicción de las cargas de trabajo en los nodos.

GRAFO DEL SISTEMA

El objetivo del particionador en la plataforma es proveer una cama de pruebas versátil y escalable con dos finalidades: probar diferentes algoritmos de particionamiento y aplicarlos a diferentes implementaciones de un circuito. Con esto en mente, se creó un particionador consistente en dos bloques: el grafo del circuito a particionar y el algoritmo de particionamiento a usar.

Sin necesidad de pasar por la etapa anterior, en este bloque se obtiene el grafo que describe al sistema primero de forma abstracta (parecido a una matriz de adyacencia) y después de forma grafica para el mejor entendimiento del sistema a ser particionado.

En esta fase, se utiliza la función del analizador anterior para encontrar los nodos pero también se implementaron funciones que detectan las relaciones entre ellos, lo que dará lugar a los arcos o conexiones entre ellas en el grafo (Véase Figura 4.17).

La forma abstracta que describe a este grafo es una tabla que primero lista los nodos y después enlista cada arco, uno en cada línea, para dar lugar a las conexiones.

Opcionalmente, el programa puede leer el archivo de estadística para agregar los números como pesos en cada arco según corresponda.

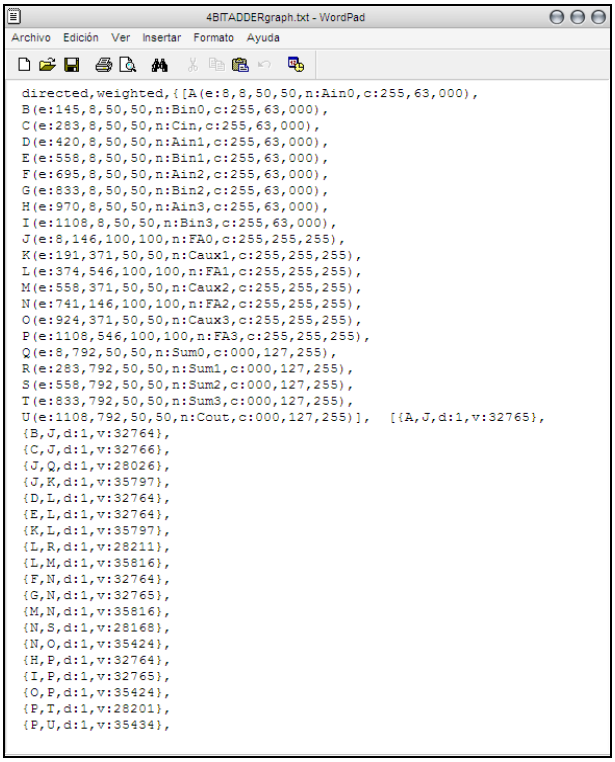

Figura 4.17. Tabla de descripción del grafo.

Para visualizar esta tabla de forma gráfica, se toma en cuenta el Applet de CPMP Tools llamado Vertex-Edge. Disponible en http://www.wmich.edu/cpmp/. Vista del Grafo: Figura 4.18.

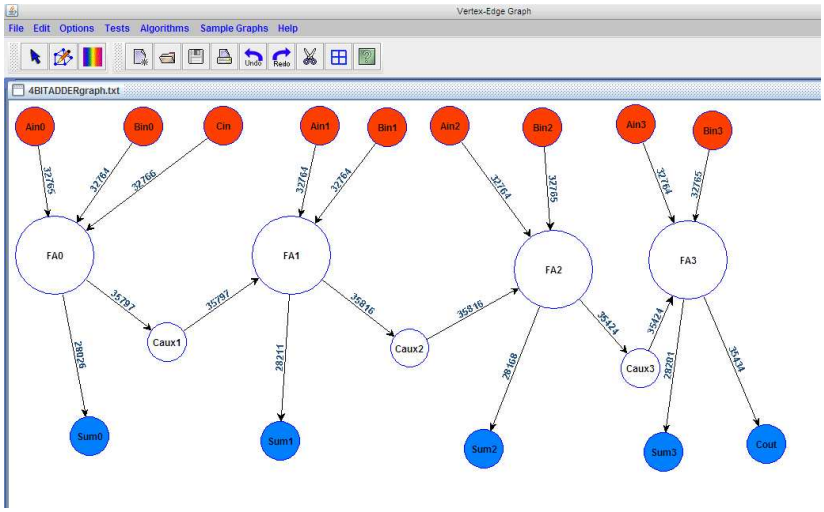

Figura 4.18. Construcción del grafo en forma grafica.

PARTICIONAMIENTO

Ahora bien, ya que se tiene un enfoque de la actividad efectiva del sistema a verificar, deben formarse las particiones que permitirán distribuir la simulación.

Aunque todas las etapas anteriores son importantes, esta es la crítica, pues aquí se harán los cortes y se crearan los módulos ejecutables por separado para su posterior distribución.

En el programa que se encarga de esta parte, se ha implementado un algoritmo que identifica los sub módulos de los que está compuesto el sistema y los considera como una partición. Para esto, debe conocer las entradas y salidas de cada uno de estos sub módulos, saber que tipos de datos maneja y además a cuales les debe entregar información.

Este programa "desensambla" en sub módulos independientes pero que se comunican entre sí mediante colas. La información obtenida por esta etapa de descomposición ayudará a "reconstruir" el diseño mapeando cada sub módulo a una entidad de ejecución por separado, con la idea de que se ejecuten concurrentemente. (Véase Figura 4.19).

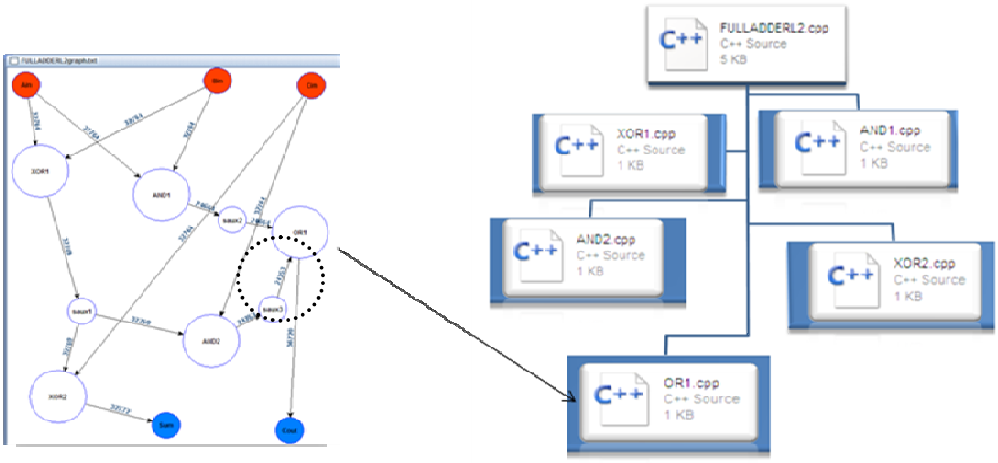

Figura 4.19. Creación de sub módulos.

Después de esta etapa, un Script se encargará de hacer una compilación de los archivos en C++ usando la biblioteca de SystemC para todas y cada una de las particiones, (Véase Figura 4.20). Esto generará varios archivos ejecutables (el mismo número de particiones) teniendo así listos los elementos que serán distribuidos por el motor de ejecución paralela.

La paralelización de estos procesos es lo que dará lugar a la aceleración en los tiempos de simulación, sin embargo, como pasa en cualquier programa distribuido, la sincronización es fundamental para asegurar un correcto procesamiento de los datos.

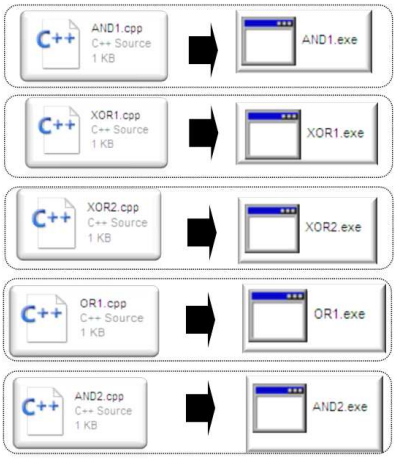

Figura 4.20. Compilación de los archivos en C++.

4.2.1.1 Características de los Equipos de Cómputo que Conforman la Malla para la Aplicación (Sumador de 1 Bit con Acarreo).

En la tabla 4.2 se describen las características de los equipos que conformaron la Malla sobre la cual, se realizaron las pruebas.

Tabla 4.2. Características de las computadoras con los que se evaluó el rendimiento del sistema utilizando la aplicación (sumador de 1 bit con acarreo).

IP	CARACTERISTICAS	DESEMPEÑO DE TRABAJO COMO:
	-Velocidad de Procesador: **2 GHz.** -Tamaño de la RAM: **2.00 GB.** -Espacio libre en la unidad C: **23.8 GB.** -Sistema Operativo: **Windows Vista Home Premium.**	**Usuario y Maestro**
148.204.64.209	-Velocidad de Procesador: **3 GHz.** -Tamaño de la RAM: **1.00 GB.** -Tamaño de la unidad C: **54.6 GB.** -Sistema Operativo: **Windows XP Profesional SP2.**	**Trabajador**
148.204.64.206	-Velocidad de Procesador: **3 GHz.** -Tamaño de la RAM: **1.00 GB.**	**Trabajador**

	-Tamaño de la unidad C: **46.9 GB.** -Sistema Operativo: **Windows XP Profesional SP2.**	
148.204.64.203	-Velocidad de Procesador: **3 GHz.** -Tamaño de la RAM: **1.00 GB.** - Tamaño de la unidad C: **66.5 GB.** -Sistema Operativo: **Windows XP Profesional SP2.**	**Trabajador**
148.204.64.205	-Velocidad de Procesador: **2.53 GHz.** -Tamaño de la RAM: **1.00 GB.** -Tamaño de la unidad C: **97.6 GB.** -Sistema Operativo: **Windows XP Profesional SP2.**	**Trabajador**
148.204.64.199	-Velocidad de Procesador: **2.53 GHz.** -Tamaño de la RAM: **1.00 GB.** -Tamaño de la unidad C: **149 GB.** -Sistema Operativo: **Windows XP Profesional SP2.**	**Trabajador**
148.204.64.197	-Velocidad de Procesador: **3 GHz.** -Tamaño de la RAM: **1.00 GB.** -Tamaño de la unidad C: **66.5 GB.** -Sistema Operativo: **Windows XP**	**Trabajador**

4.2.1.2 Características de la Aplicación (Sumador de 1 bit con Acarreo)

En la Tabla 4.3 muestra la descripción de la aplicación utilizada para una de las pruebas realizadas al sistema, además se especifica el número de operaciones realizadas por este.

Tabla 4.3. Características de la aplicación (sumador de 1 bit con acarreo).

NOMBRE	CANTIDAD DE OPERACIONES	DESCRIPCIÓN
1.- Sumador de 1 bit con acarreo.	Vectores de prueba * número de vértices. (2^10) * 8 = 8192	Representa la simulación de un circuito digital, el cual trabaja con vectores de prueba de 2^{10}. Consta de 8 archivos ejecutables que interactúan entre sí para cumplir con su objetivo. Los nombres de los archivos son los siguientes: AIN, BIN,

		CIN XOR1, XOR2, AND1, AND2, OR.

4.2.1.3 Evaluación del Rendimiento del Sistema Utilizando la Aplicación (Sumador de 1 bit con Acarreo)

La distribución de los procesos se puede observar en la Tabla 4.4. La distribución se realizó de manera adecuada, ya que los 6 trabajadores se clasificaron con alto poder de cómputo por lo cual pueden ejecutar cualquier proceso, además de ello el sistema siempre fue asignando un nuevo proceso al trabajador que menos tenía en ejecución.

Tabla 4.4. Distribución de los procesos (sumador de 1 bit con acarreo) entre los trabajadores.

NÚMERO DE TRABAJADORES	DISTRIBUCIÓN
1	**148.204.64.205:** AIN, BIN, CIN, XOR1, XOR2, AND1,AND2, OR1
2	**148.204.64.205:** BIN, XOR1, XOR2, OR1 **148.204.64.209:** AIN, CIN, AND1, AND2
3	**148.204.64.205:** BIN, AND1, OR1 **148.204.64.209:** AIN, XOR1, AND2 **148.204.64.206:** CIN, XOR2
4	**148.204.64.205:** CIN, AND2 **148.204.64.209:** AIN, AND1 **148.204.64.206:** OR1, XOR1 **148.204.64.203:** BIN, XOR2
5	**148.204.64.205:** XOR1 **148.204.64.209:** BIN, AND2 **148.204.64.206:** AND1 **148.204.64.203:** CIN, OR1 **148.204.64.197:** AIN, XOR2
6	**148.204.64.205:** XOR1 **148.204.64.209:** BIN, OR1 **148.204.64.206:** XOR2 **148.204.64.203:** CIN **148.204.64.197:** AIN, AND2 **148.204.64.199:** AND1

En la Figura 4.21 se puede observar el rendimiento que tuvo el sistema para la simulación del sumador de 1 bits con acarreo, el cual fue particionado en 8 procesos; en dicha simulación participaron 6 trabajadores. La gráfica muestra que el tiempo de duración de la prueba incrementa entre más trabajadores haya, esto se debe a que la latencia de las comunicaciones entre trabajadores se vuelve mayor al tiempo que tarda cada proceso en realizar sus tareas. Esto se da debido a que es más pequeño el tiempo de cómputo en comparación con el tiempo existente en las comunicaciones más el tiempo de lectura y escritura de las colas.

Figura 4.21. Gráfica que muestra el rendimiento del sistema evaluado con la aplicación (sumador de 1 bit con acarreo).

4.2.2 DESCRIPCIÓN DEL DESARROLLO DE LA APLICACIÓN (LOCALIZADOR DE LAS RUTAS MÁS CORTAS)

INTRODUCCIÓN

Esta aplicación encuentra la mejor ruta que hay entre dos puntos en una ciudad. Entiéndase por mejor ruta a aquella que genere menos tiempo o que recorra menos distancia entre dos puntos. Por ejemplo, una familia que se encuentra en Milpa Alta desea saber cuál es la mejor ruta posible para ir a Tepoztlan, (encerrados en círculos azules), (Véase Figura 4.22).

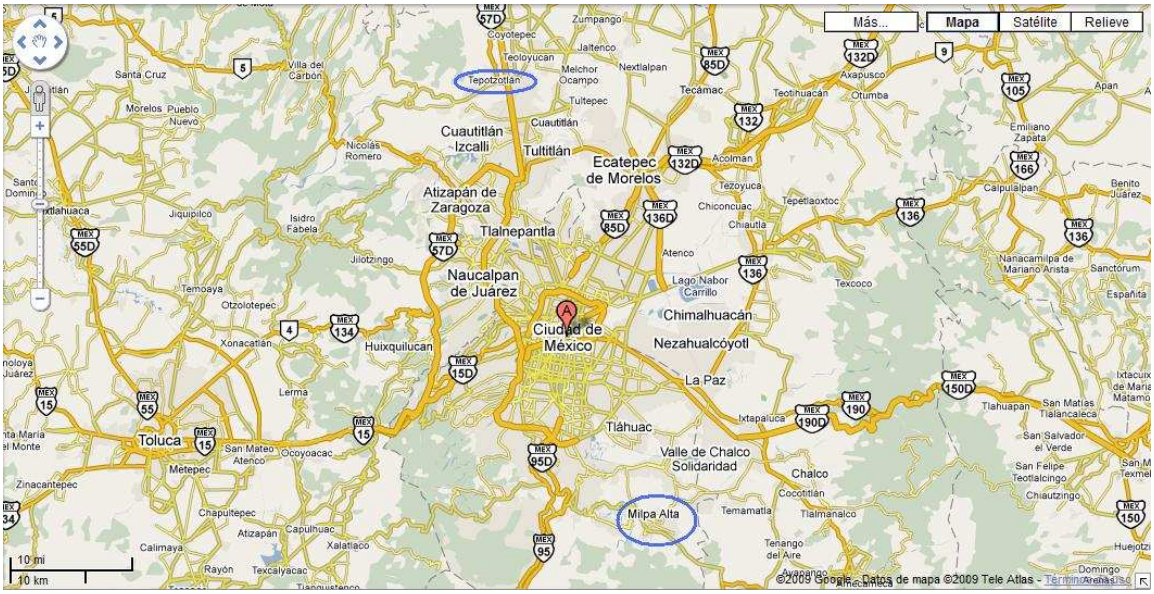

Figura 4.22. Mapa de lugares colindantes con la ciudad de México.

Para lograr lo anterior, se ha desarrollado una aplicación que pretende simular el flujo del tráfico que hay en una ciudad, por tanto pudiéramos suponer que existen localizadores GPS (Global Position System, Sistema de Posición Global) en cada uno de los coches que transitan las calles de una cierta ciudad, dichos localizadores envían la información perteneciente a cada una de las calles (velocidad con la que se puede transitar de una calle a otra o simplemente la distancia que hay entre las calles), toda esta información es almacenada en una base de datos llamada GPS.

Esta aplicación consta de 4 módulos principales:

- Creación y llenado de la tabla que almacena las distancias entre cada calle.
- División de la tabla y asignación de bloques.
- Búsqueda de las rutas más cortas.
- Unión de rutas.

CREACIÓN Y LLENADO DE LA TABLA QUE ALMACENA LAS DISTANCIAS ENTRE CADA CALLE

Para poder simular lo anterior se creó un proceso que se encarga de crear una tabla llamada "ciudad" dentro de la base de datos GPS, la cual tiene la capacidad de almacenar 64 * 64 campos,

cada uno de ellos es llenado por dicho proceso con tres valores diferentes, los cuales indican los estados transitorios que existen en las tres direcciones posibles para trasladarse de una calle a otra, los valores en cada campo tienen el siguiente significado:

- Valor 1.- Indica la información correspondiente que existe para trasladarse en dirección (hacia arriba).
- Valor 2.- Indica la información correspondiente que existe para trasladarse en dirección (hacia la derecha).
- Valor 3.- Indica la información correspondiente que existe para trasladarse en dirección (hacia abajo).

NOTA: En caso de que alguno o algunos de los valores en cada campo llegue a ser cero, estará indicando que en la dirección correspondiente no puede haber traslado. Además de ello, se ha definido que las direcciones hacia la izquierda en cada calle no sean posibles.

DIVISIÓN DE LA TABLA Y ASIGNACIÓN DE BLOQUES

Una vez que la tabla contiene la información del estado transitorio de una cierta ciudad, un nuevo proceso llamado “ActualizaBase” se encarga de generar particiones a la tabla en un determinado número de bloques (especificado por un usuario), estas particiones se dan porque se pretende analizar los tiempos de cómputo cuando la ciudad se divide en bloques de tamaño (1, 2, 4, 8, 16). El proceso encargado de realizar esta división, toma cada bloque generado y asigna los datos dentro de este a una fila de un archivo de texto, por tanto habrá tantas filas con datos como bloques generados en la partición de la tabla ciudad. De esta manera se logra que el sistema de Computación en Malla pueda distribuir cada bloque de datos.

BÚSQUEDA DE LAS RUTAS MÁS CORTAS

El proceso que se encarga de buscar la ruta más corta se llama “Localizador”, este toma los datos de su cola (archivo de texto) de entrada, dicha cola se compone de dos filas las cuales se describen a continuación:

- Fila 1. recibe lo siguiente: el número que tiene el bloque, tamaño del bloque, total de bloques generados, además de esto, información que indica si el nodo origen y/o destino se encuentran dentro del bloque y por último los datos del bloque.
- Fila 2.- recibe información que indica cual es el nodo origen y la ruta acumulada en caso de haberla.

Primeramente "Localizador" toma la información de la fila 1, a partir de que obtiene los datos construye el grafo de comunicaciones donde se indica que nodo tiene comunicación con otros nodos y además los costos (distancias entre cada conexión de calles) de estas comunicaciones. Una vez generado el grafo se procede a obtener la mejor ruta de todos los posibles orígenes con respecto a todos los posibles destinos.

Para ello se obtiene una representación de todos los nodos en forma de matriz para poder obtener a los posibles orígenes y a los posibles destinos. Existen tantos orígenes y tantos destinos como número de filas en la matriz de nodos, por tanto si la tabla que contiene los datos de la ciudad es de tamaño 64 * 64 para cada partición que se genere existirán 64 posibles orígenes y 64 posibles destinos, esto es así ya que las particiones se realizan en forma horizontal y siempre conservan el mismo tamaño verticalmente.

Una vez obtenidos los destinos y los orígenes, el proceso procede obtener la mejor ruta posible de cada uno de los orígenes con respecto a cada uno de los destinos, para ello utiliza el algoritmo de Dijkstra (conocido como el algoritmo de caminos mínimos o de la ruta más corta).

Entonces para cada nodo origen, obtiene solo la mejor ruta con respecto a todos los nodos destinos, de tal manera que al finalizar su ejecución genera una lista de rutas finales, donde se indica cual es la mejor ruta para los posibles nodos orígenes con respecto al nodo destino que genere la ruta menos costosa.

NOTA: Se realiza el cálculo de la mejor ruta para cada posible nodo origen ya que si la tabla ciudad fue particionada en más de un bloque, los bloques posteriores al inicial no saben cual podrá ser su nodo origen, por tanto siempre se calculan las mejores rutas para todos los posibles nodos orígenes.

UNIÓN DE RUTAS

Ya que se ha obtenido la información de la fila 2 de la cola de entrada del proceso "Localizador", verifica si en esta ella se ha indicado un nodo destino, en caso de ser afirmativo, procede a buscar en la lista de rutas finales a aquella que concuerde tanto el nodo origen como el destino. En caso de que no se haya especificado un nodo destino se obtiene de la lista de rutas finales a aquella que concuerde únicamente con el nodo origen obtenido en la fila 2.

Por último el proceso "Localizador" escribe en su cola de salida lo siguiente:

- Si la tabla ciudad no fue particionada más que en un solo bloque de datos, escribe la mejor ruta obtenida para llegar del nodo origen al nodo destino indicado por el usuario.
- Si la tabla ciudad fue particionada en más de un bloque de datos, escribe el nodo origen que tendrá el siguiente bloque de datos y la mejor ruta que obtuvo. De esta manera los demás localizadores que computen a los otros bloques de datos podrán saber cual nodo será el origen para su bloque y así irán uniendo la mejor ruta que hayan obtenido, a la que otros localizadores ya obtuvieron.

Al final, el "Localizador" que haya procesado el último bloque de datos escribirá la ruta final en su cola de salida para que el sistema de Computación en Malla presente dicha ruta al usuario.

OPERACIONES QUE REALIZA EL PROCESO LOCALIZADOR DE LAS RUTAS MÁS CORTAS

El tamaño de la tabla ciudad indica cuantos nodos (uniones entre calles) existen en la ciudad, para este caso la tabla tiene un tamaño de 64 * 64 lo que nos da un total de nodos igual a 4096, para cada nodo, el proceso llamado "Localizador" tiene que ir verificando la mejor ruta en las tres direcciones que se puede avanzar de un nodo hacia otro, lo que genera un total de operaciones igual a 4096 * 3 = 12288, además de ello este proceso tiene que realizar lo anteriormente mencionado para cada posible origen hacia cada posible destino. Para este caso existen 64 posibles orígenes y 64 posibles destinos, por tanto en total se realizan un total de operaciones igual a 64 * 12288 * 64 = 50331648.

4.2.2.1 Características de los Equipos de Cómputo que Conforman la Malla para la Aplicación (Localizador de las Rutas más Cortas).

En la tabla 4.5 se describen las características de los equipos que conformaron la Malla sobre la cual, se realizaron las pruebas para la aplicación (Localizador de las rutas más cortas).

Tabla 4.5. Características y recursos de las computadoras que conformaron la Malla.

IP	CARACTERISTICAS	DESEMPEÑO DE TRABAJO COMO:
148.204.64.202	-Velocidad de Procesador: **Core 2 Duo a 2.0 GHz.** -Tamaño de la RAM: **2.00 GB.** -Espacio libre en la unidad C: **23.8 GB.** -Sistema Operativo: **Windows Vista Home Premium.**	**Usuario, Maestro y Trabajador**
148.204.64.199	-Velocidad de Procesador: **Core 2 Duo a 2.53 GHz.** -Tamaño de la RAM: **1.00 GB.** -Tamaño de la unidad C: **149 GB.** -Sistema Operativo: **Windows XP Profesional SP2.**	**Trabajador**
148.204.64.205	-Velocidad de Procesador: **Core 2 Duo a 2.53 GHz.** -Tamaño de la RAM: **1.00 GB.** -Tamaño de la unidad C: **97.6 GB.** -Sistema Operativo: **Windows XP Profesional SP2.**	**Trabajador**
148.204.64.206	-Velocidad de Procesador: **Core 2 Duo a 2.50 GHz.** -Tamaño de la RAM: **4.00 GB.** -Tamaño de la unidad C: **46.9 GB.** -Sistema Operativo: **Windows Vista Bussines.**	**Trabajador**
148.204.64.203	-Velocidad de Procesador: **3 GHz.** -Tamaño de la RAM: **1.00 GB.** - Tamaño de la unidad C: **66.5 GB.** -Sistema Operativo: **Windows XP Profesional SP2.**	**Trabajador**
148.204.64.209	-Velocidad de Procesador: **3 GHz.** -Tamaño de la RAM: **1.00 GB.** -Tamaño de la unidad C: **54.6 GB.**	**Trabajador**

	-Sistema Operativo: **Windows XP Profesional SP2.**	
148.204.64.197	-Velocidad de Procesador: **3 GHz.** -Tamaño de la RAM: **1.00 GB.** -Tamaño de la unidad C: **66.5 GB.** -Sistema Operativo: **Windows XP**	**Trabajador**

4.2.2.2 Evaluación del Rendimiento del Sistema Utilizando la Aplicación (Localizador de las Rutas más Cortas)

La distribución de los procesos se puede observar en la Tabla 4.6. Las siete computadoras participantes en las pruebas fueron clasificados con la categoría de "alto poder de cómputo", además de ello los procesos en los que se dividió a la aplicación .fueron clasificados como procesos pesados, por tanto se realizo la siguiente distribución:

Tabla 4.6. Distribución de los procesos entre los trabajadores para una ciudad de tamaño de 64 * 64 y 32 * 32.

NÚMERO DE POCESOS	DISTRIBUCIÓN
1	**148.204.64.202:** ActualizaBase, Localizador1 **148.204.64.199:** **148.204.64.205:** **148.204.64.206:** **148.204.64.197:** **148.204.64.203:** **148.204.64.209:**
2	**148.204.64.202:** ActualizaBase, Localizador2 **148.204.64.199:** Localizador1 **148.204.64.205:** **148.204.64.206:** **148.204.64.197:** **148.204.64.203:** **148.204.64.209:**
4	**148.204.64.202:** ActualizaBase, Localizador4 **148.204.64.199:** Localizador1 **148.204.64.205:** Localizador2 **148.204.64.206:** Localizador3 **148.204.64.197:**

	148.204.64.203: **148.204.64.209:**
8	**148.204.64.202:** ActualizaBase, Localizador7 **148.204.64.199:** Localizador1, Localizador8 **148.204.64.205:** Localizador2 **148.204.64.206:** Localizador3 **148.204.64.197:** Localizador4 **148.204.64.203:** Localizador5 **148.204.64.209:** Localizador6
16	**148.204.64.202:** ActualizaBase, Localizador7, Localizador14 **148.204.64.199:** Localizador1, Localizador8, Localizador15 **148.204.64.205:** Localizador2, Localizador9, Localizador16 **148.204.64.206:** Localizador3, Localizador10 **148.204.64.197:** Localizador4, Localizador11 **148.204.64.203:** Localizador5, Localizador12 **148.204.64.209:** Localizador6, Localizador13

En la tabla 4.6 podemos observar que se realizo una distribución adecuada ya que el sistema siempre fue ubicando los procesos en la computadora que menos procesos tenía en ejecución.

NOTA: El proceso llamado "ActualizaBase" no formó parte del conteo de procesos que se iban a distribuir, ya que este proceso siempre era almacenado en la computadora con la dirección IP "148.204.64.202", debido a que en esta se encuentra la base de datos donde el proceso tiene que obtener los datos.

EVALUACIÓN DEL RENDIMIENTO DEL SISTEMA DONDE LA TABLA "CIUDAD" TIENE UN TAMAÑO DE 64 * 64

En la Figura 4.23 se puede observar el rendimiento que tuvo el sistema para la aplicación Localizador de las rutas más cortas, donde el tamaño de la tabla de datos "ciudad" es de 64 * 64 campos.

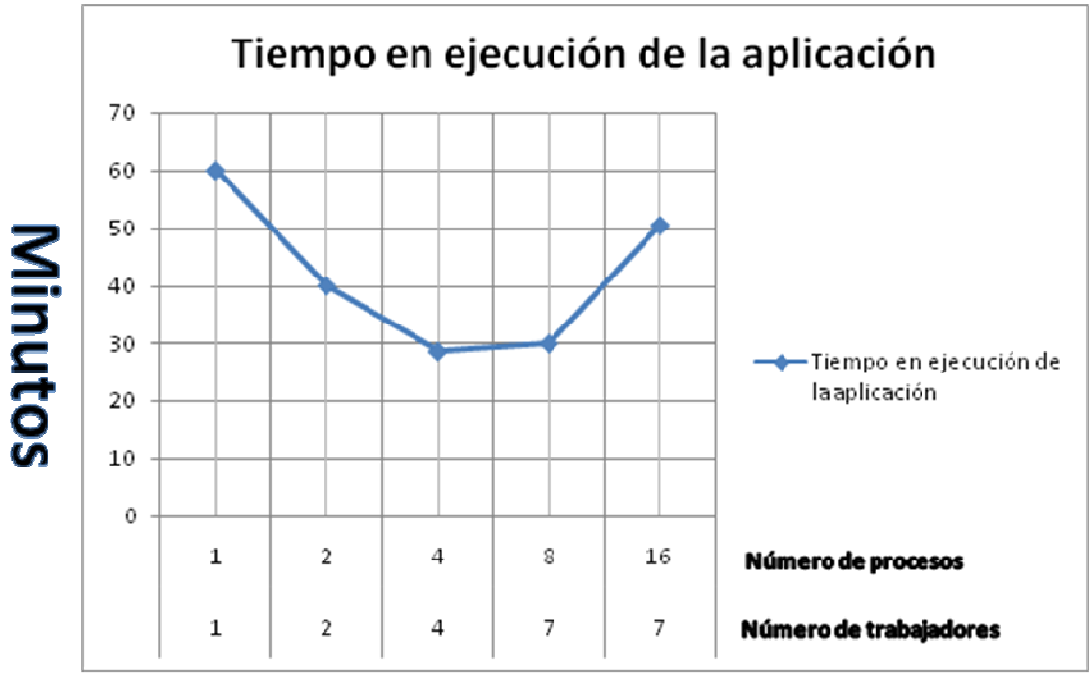

Figura 4.23. Gráfica que muestra el rendimiento del sistema evaluado con la aplicación (Localizador de las rutas más cortas) con tamaño de la tabla de datos de 64 * 64.

Como se puede observar en la figura 4.23, el tiempo de ejecucion disminuye cuando se va aumentando el número de trabajadores, esto hasta llegar a cuatro trabajadores (Veasé Tabla 4.5). Al inrementar el número de trabajadores a 7, el tiempo de ejeución aumenta, esto se debe a que los primeros cuatro trabajadores tienen procesadores core 2 Duo a 2.53 GHz y los últimos tres tienen procesadores de 1 solo core a 3GHz, por lo cual como se puede observar en la gráfica, al participar los últimos tres trabajadores (Veasé Tabla 4.5), los tiempos comenzaron a aumentar generando un cuello de botella debido a que les demoraba mucho más tiempo ejecutar los proceso a estos trabajadores en comparación con los cuatro primeros trabajadores.

Para demostrar que los últimos tres trabajadores generaron un cuello de botella, se compararon los tiempos de ejeución para 8 y 16 procesos ejecutandolos solo en los primeros cuatro trabajadores con respecto a la ejecución de 8 y 16 procesos ejecutados en los 7 trabajadores donde participan los 3 trabajadores con menores recursos. Esto se puede observar en la Figura 4.24.

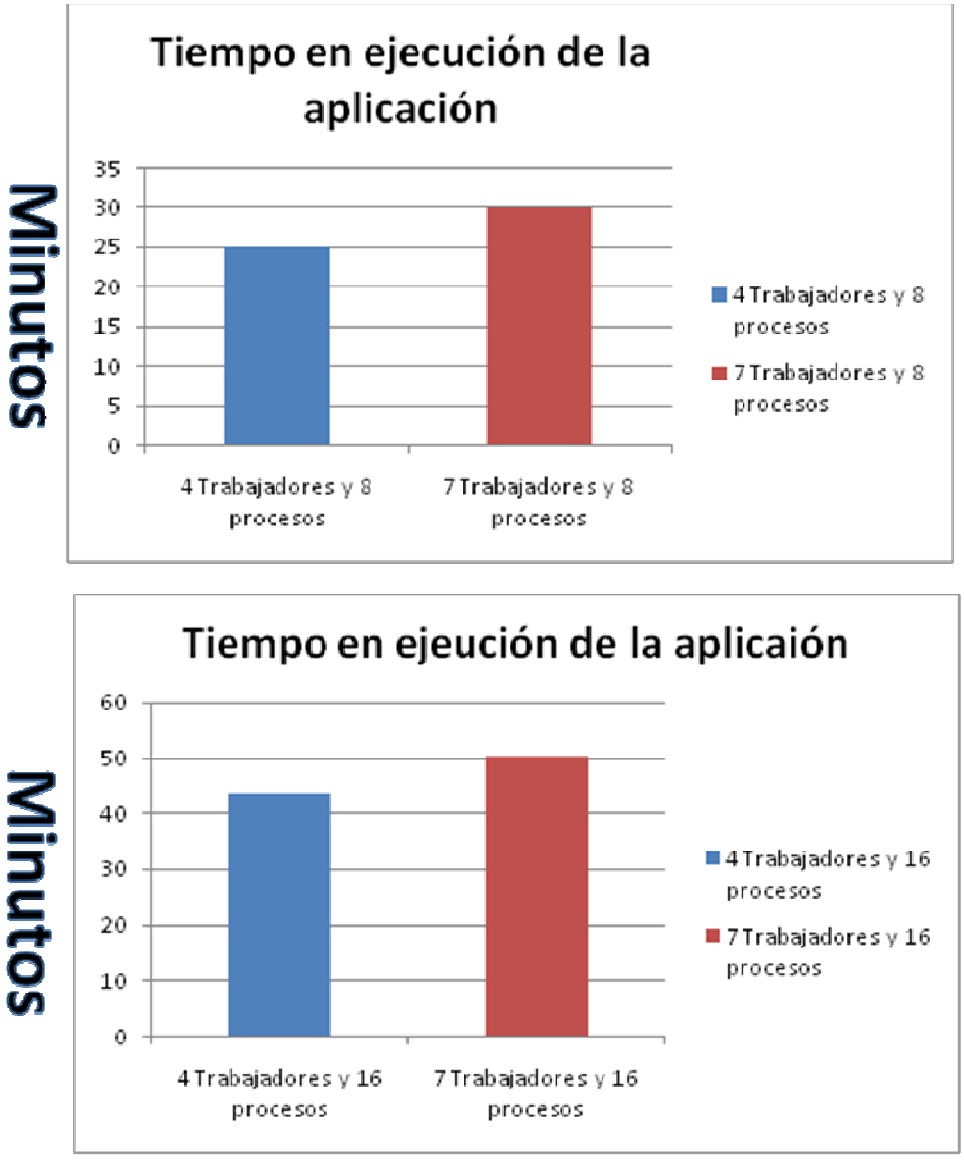

Figura 4.24. Comparación de tiempos de ejecución para 8 y16 procesos en 4 trabajadores, asi mismo 8 y 16 procesos en 7 trabajadores teniendo un tamaño en la tabla de datos de 64* 64.

EVALUACIÓN DEL RENDIMIENTO DEL SISTEMA DONDE LA TABLA "CIUDAD" TIENE UN TAMAÑO DE 32 * 32

En la Figura 4.25 se puede observar el rendimiento que tuvo el sistema para la aplicación Localizador de las rutas más cortas, donde la el tamaño de la tabla de datos "ciudad" tiene un tamaño de 32 * 32.

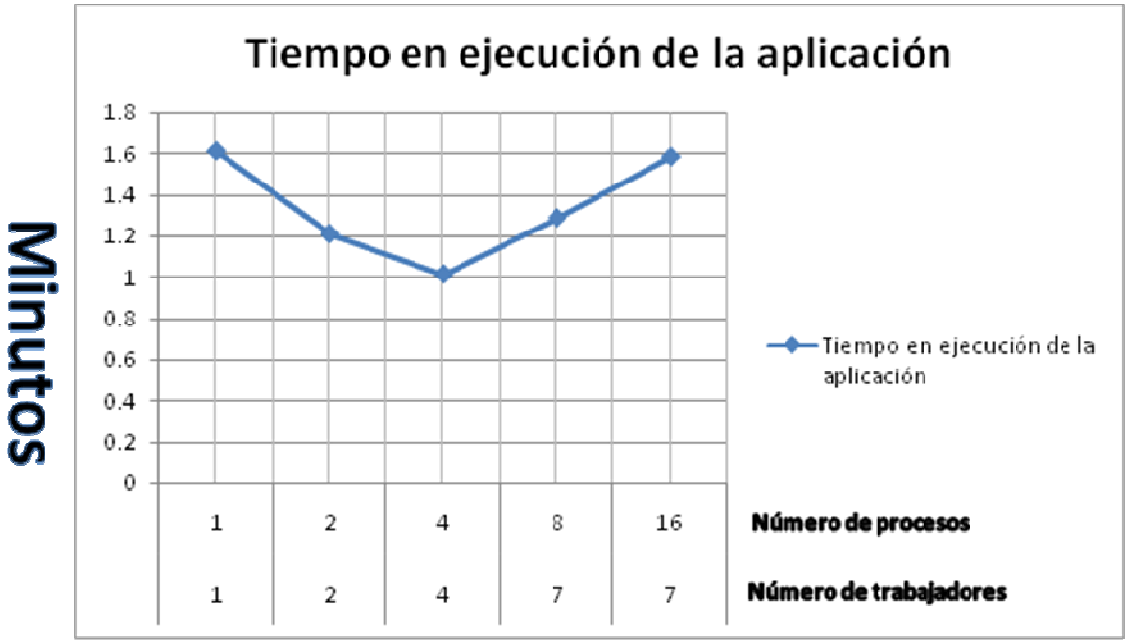

Figura 4.25. Gráfica que muestra el rendimiento del sistema, evaluado con la aplicación (Localizador de las rutas más cortas) con tamaño de la tabla de datos de 32 * 32.

Al igual que en la prueba anterior, se observa en la Figura 4.25 que al participar los 7 trabajadores el tiempo de ejecución de la aplicación comienza a aumentar, también se debe a que los últimos tres trabajadores generan un cuello de botella.

Nuevamente para demostrar que los últimos tres trabajadores generaron dicho cuello de botella, se compararon los tiempos de ejeución para 8 y 16 procesos ejecutandolos solo en los primeros cuatro trabajadores con respecto a la ejecución de 8 y 16 procesos ejecutados en los 7 trabajadores donde participan los 3 trabajadores con menores recursos. Esto se puede observar en la Figura 4.26.

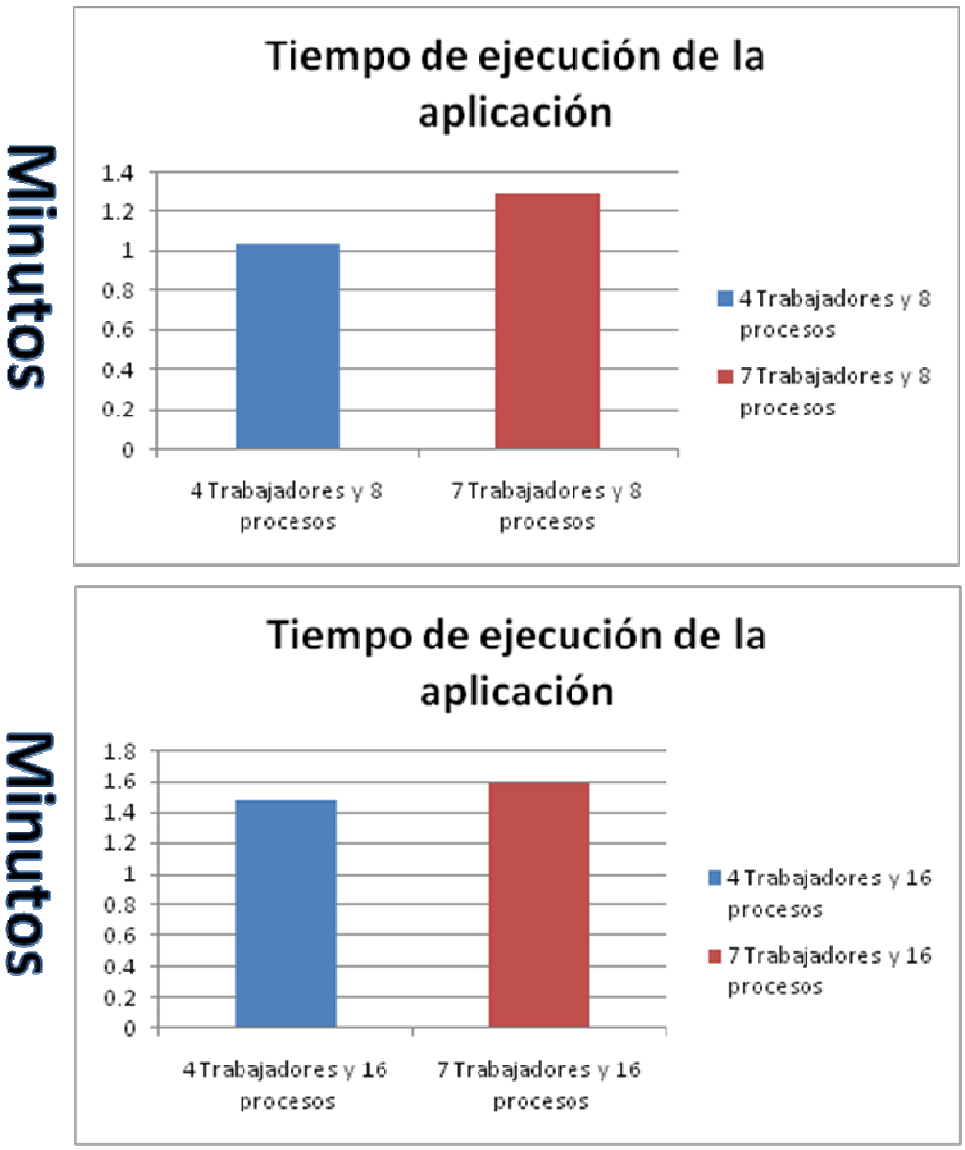

Figura 4.26. Comparación de tiempos de ejecución para 8 y16 procesos en 4 trabajadores, asi mismo 8 y 16 procesos en 7 trabajadores teniendo un tamaño en la tabla de datos de 32*32.

De esta forma se demuestra que para aplicaciones que son demandantes en cuanto a cantidad de cómputo, si se generan particiones de estas aplicaciones, el sistema de Computación en Malla puede realizar una la distribución de los procesos que conformen a una aplicación de tal manera que se logre la disminucion del tiempo de cómputo para dichas aplicaciones, siempre y cuando las computadoras que pertenezcan a la Malla sean homogenéas en cuanto a sus reursos.

Cuando las computadoras que forman la Malla son heterogenéas en cuanto a recursos, el sistema de Computación en Malla realiza la distribución de procesos de la manera más eficaz, tomando en siempre en cuenta las capacidades de cómputo de cada computadora, desgraciadamente se puede dar el caso en que el tiempo de ejecución de las aplicaciones llegue a ser mayor cuando se incrementa el número de computadoras participantes si estas no son homogenéas en cuanto a recursos.

Capítulo 5

Conclusiones y Trabajos a Futuro

Se muestran las conclusiones generadas para este trabajo de investigación, además de que se describen los puntos que se han definido como trabajo a futuro.

El contenido de los temas presentados en este capítulo incluye:

- Conclusiones.
- Trabajos a futuro.
- Publicaciones generadas durante el periodo de estudios de la maestría.

5.1. CONCLUSIONES

Este proyecto de investigación se enfocó a crear un sistema que trabajará a nivel middleware, ofreciendo un soporte para que aquellos laboratorios virtuales pertenecientes a las áreas de ingeniería que tengan la necesidad de ejecutar aplicaciones muy demandantes en cuanto a poder de cómputo lo puedan realizar en forma distribuida sobre una Malla conformada por computadoras de escritorio las cuales pueden ser o no heterogéneas; siempre y cuando tengan la capacidad de entregarle al sistema un conjunto de archivos ejecutables(procesos) los cuales conformen a una aplicación ya particionada y además un archivo de configuración con un formato ya establecido donde se especifique el grafo de comunicaciones entre los procesos.

Se creó una arquitectura la cual consta de tres elementos principales: un sistema que se encarga de crear una plataforma estática para configurar al sistema de tal forma que se logre la ejecución de una aplicación en forma distribuida, ya que esta se encarga de crear un contenedor de procesos el cual contiene la información necesaria para que cada uno de los elementos principales que intervienen en esta arquitectura puedan realizar su trabajo en forma adecuada, teniendo como objetivo que el contenedor sea enviado entre las partes principales del sistema donde cada una de ellas pueda obtener datos, información y los procesos a ejecutar, según lo requiera cada uno de los elementos principales. El segundo elemento en la arquitectura es un maestro que se encarga de obtener los recursos de todas aquellas computadoras que formen parte de la Malla, logrando utilizar la capacidad de procesamiento que tiene cada una de ellas en su beneficio para realizar la ejecución de procesos que pertenecen a aplicaciones intensivas, para lo cual realiza el balanceo de las cargas de trabajo y la distribución de todos aquellos procesos que conformen una aplicación entre la múltiples unidades de ejecución que tiene a su disposición, basándose en los requerimientos de cada proceso y en los recursos disponibles en cada unidad de ejecución. Como tercer elemento se tienen múltiples unidades de ejecución llamas trabajadores, siendo estos los que se encargan de realizar la ejecución de los procesos y lo que se encargan de solventar la dependencia de datos que exista entre ellos, logrando con ello evitar una saturación de comunicaciones en el maestro, ya que los trabajadores le evitan realizar todas aquellas comunicaciones que tengan que ver con el transporte de resultados, esto se debe gracias a que se generan comunicaciones en forma punto a punto entre trabajadores e interfaz de usuario para solventar la dependencia de datos entre procesos. Cada trabajador se encarga de transportar los resultados que generen los procesos que tienen en ejecución a un cierto destino.

Siendo esta una diferencia con respectos a los sistemas mostrados en el estado del arte, ya que en aquellos sistemas el maestro es el encargado de realizar el transporté de resultados de un lugar a otro.

En base a las pruebas realizadas se demostró que el sistema en términos de funcionalidad cumple con los objetivos planteados, ya que este se encarga de ejecutar aplicaciones en forma distribuida realizando la planificación y distribución de procesos de la manera más conveniente para cada una de las aplicaciones generándoles así un ambiente de ejecución particular.

El sistema en términos de escalabilidad permite agregar nuevas computadoras para que formen parte de la Malla, de igual manera el tiene la capacidad de descartar a todas aquellas computadoras que por alguna razón se encuentren desconectadas de la red en el momento en que se va a realizar la distribución de procesos, aunado a esto, una de las principales ventajas de este sistema es que tiene la capacidad de trabajar con computadoras heterogéneas en cuanto a recursos.

En cuanto a la forma de almacenamiento de resultados por medio de archivos de texto (colas) se demostró que funcionan de forma correcta, ya que le permite al sistema almacenar dichos resultados de manera que siempre se vayan procesando en el orden correcto evitando que haya un desfasamiento de datos, lo cual podría generar resultados incorrectos.

Se desarrolló una interfaz de usuario con la cual se pueden observar los resultados que vayan generando los procesos que arrojan resultados finales durante el tiempo que tarde su ejecución, además permite observar todos aquellos eventos registrados en la bitácora.

Se creó un proceso encargado de verificar los recursos de cada uno de los trabajadores como son: RAM libre, velocidad de la RAM, espacio en disco duro libre, velocidad y número de microprocesadores. De esta manera se puede verificar el estado de cada trabajador antes de enviarle más trabajo ayudando con ello a realizar una mejor planificación.

Se desarrolló un modelo de paso de mensajes entre procesos que trabaja en forma punto a punto, logrando con ello realizar comunicaciones directas y más eficientes para el transporté de datos entre trabajadores.

El rendimiento que se obtuvo para la prueba (sumador de 1 bit con acarreo), basados en las pruebas realizadas, indican que a mayor número de trabajadores, el tiempo que tarda la simulación en terminar es mayor. Esto se debe a que la cantidad de cómputo que realiza cada proceso es considerablemente pequeña en comparación con el tiempo en que se genera en cada comunicación entre procesos, lo cual deja a la vista que a mayor número de trabajadores crece la latencia de comunicaciones de tal manera que al sistema le toma más tiempo terminar dicha simulación. Si los procesos generarán una mayor cantidad de cómputo se podrá esconder la latencia de las comunicaciones entre los trabajadores, logrando con ello una disminución en el tiempo de las simulaciones entre mayor número de trabajadores existentes.

Para la aplicación (localizador de las rutas más cortas), la cual se utilizo para medir el rendimiento del sistema, mostró que al ir aumentando el número de trabajadores, los tiempos de ejecución de la aplicación disminuían siempre y cuando las computadoras participantes eran homogéneas en cuanto a recursos computacionales, con lo cual se pudo comprobar el sistema de Computación en Malla ayuda a generar un mejor rendimiento a la hora de hacer cómputo distribuido sobre aplicaciones demandantes en cuanto a recursos, en comparación con el rendimiento obtenido cuando el cómputo se realizo en una sola computadora aislada.

Cuando las computadoras que forman la Malla no son homogenéas en cuanto a recursos, se puede dar el caso en que el tiempo de ejecución de las aplicaciones llegue a ser mayor al aumentar el número de computadoras si comienzan a participar computadoras con recursos inferiores a las otras computadoras.

5.2. TRABAJOS A FUTURO

Como trabajo a futuro se plantean los siguientes puntos para mejorar la funcionalidad del sistema:

- La creación de nuevos modelos de planificación, distribución y balanceo de cargas de trabajo, se plantea que tales modelos podrían desarrollar técnicas inteligentes para fortalecer el performance y la funcionalidad del sistema.

- Encontrar mejores mecanismos de almacenamiento de resultados, por ejemplo a través de estructuras de memoria, ya que esto agilizaría la lectura de los resultados.
- La caracterización y eliminación de cuellos de botella (latencia de comunicaciones entre procesos).
- Brindarle robustez al sistema haciéndolo tolerante a fallos, además de que tenga la capacidad de identificar donde se ha producido un error, recuperarse de este y continuar con la ejecución de la aplicación.
- Unificación del sistema Grid Computing con el sistema de particionamiento de procesos.

5.3. Publicaciones Generadas Durante el Periodo de Estudios de la Maestría

- "Alternativa tecnológica para apoyar a comunidades ubicadas en la última milla, utilizando la red eléctrica como medio de comunicación", 3er Congreso Internacional: Tendencias Tecnológicas en Computación 2007.
- "Proyecto para gestionar una base de datos que administre información de comunidades ubicadas en la última milla", 3er Congreso Internacional: Tendencias Tecnológicas en Computación 2007.
- "Instalación de OPEN SUSE LINUX 10.3", Festival Latinoamericano de Instalación de Software Libre 2008.

REFERENCIAS BIBLIOGRÁFICAS

[1] CANDIA, Alejandra. Educación virtual ¿Una alternativa a la educación tradicional?, [en línea], [Coloquio Universidad Torcuato Di Tella]. Noviembre 2000. Disponible en WWW: http://www.utdt.edu/eduforum/ensayo10.htm [Consulta: 09 Junio 2009].

[2] L. Rosado, J.R. Herreros. Nuevas aportaciones didácticas de los laboratorios virtuales y remotos en la enseñanza de la Física. Conferencia Internacional sobre multimedia y tecnologías de la comunicación y la información en la educación, Lisbon Portugal, Abril 2009. 5p.

[3] Lijun Li, Hai Huang and Carl Tropper. DVS: An Object-Oriented Framework for Distributed Verilog Simulation. Conferencia simulación paralela y distribuida, Universidad McGill, Montreal Canada, 2003. 8p. ISBN: 0-7695-1970-9.

[4] Francis L. Chan, Mark D. Spiller, A. Richard Newton. WELD – An environment for Web-Based Electronic Design. 35 conferencia anual sobre automatización del diseño, San Francisco, California, Estados Unidos. 1998. 146 – 151p. ISBN: 0-89791-964-5.

[5] B. Balamuralithara. Virtual Laboratories in Engineering Education: The Simulation Lab and Remote Lab. Cyberjaya, Malaysia. 2007

[6] Blaise Barney. OpenMP C and C++ Application Program Interface. [en línea], Marzo 2002. Disponible en WWW: http://www.openmp.org/mp-documents/cspec20.pdf. [Consulta: 09 Junio 2009].

[7] Wikipedia. Historia del Grid Computing. [en línea], Última modificación Noviembre 2008. Disponible en WWW: http://en.wikipedia.org/wiki/Grid_computing#History. [Consulta: 02 Marzo 2009].

[8] Edward Ashford Lee and Daving G. Messerchmi'tt. Static Scheduling of Synchronous Data Flow Programs for Digital Signal Processing, Washington DC. USA, Enero 1987. Volumen 36. 24-35p. ISSN: 0018-9340.

[9] Akshay Luther, Rajkumar Buyya, Rajiv Ranjan, and Srikumar Venugopal. Alchemi: A .NET-based Enterprise Grid Computing System, 6th International Conference on Internet Computing (ICOMP'05), Las Vegas, USA, June 2005. CSREA Press, Las Vegas, USA.

[10] Chao Jin, Rajkumar Buyya. A dataflow model for .NET-based Grid Computing systems, Laboratorio de Grid Computing y sistemas distribuidos, departamento de informática e ingeniería de software de la Universidad de Melbourn Australia. 2007.

[11] Xingchen Chu, Krishna Nadiminti, Chao Jin, Srikumar Venugopal, Rajkumar Buyya. Aneka: Next-Generation Enterprise Grid Platform for e-Science and e-Business Applications,

Tercera conferencia internacional IEEE sobre e-Ciencia y Grid Computing, Bangalore, India, IEEE CS Press, Los Alamitos, CA, USA. Diciembre 2007. 9p.

[12] Tiberiu STEF-PRAUN, Benjamin CLIFFORD, Ian FOSTER, Uri HASSON, Michael Hategan, Steven L. Small , Michael Wilde, Yong Zhao. Accelerating Medical Research using the Swift Workflow System, Universidad de Chicago, USA. 2007. 10p.

[13] Q. Peng D.P. Schissel M. Thompson I. Foster M. Greenwald D. McCune K. Keahey, T. Fredian. The Anatomy of the Grid: Enabling Scalable Virtual Organizations, Primer IEEE/ACM Simposio sobre Computación de Clúster y Grids. Enero 2001, 1-4p. ISSN: 0302-9743.

[14] J.Ñick S. Tuecke I. Foster, C. Kesselman. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, Foro global sobre Grid, USA. Junio 2002. 31p.

[15] T. Durniak P. Herman J. Karuturi C.Woods C. Gilman J. Barry, M. Aparicio. Enterprise Distributed Object Computing Workshop, Procedente de IEEE, USA. Noviembre 1998.

[16] Jay Unger. The Physiology of the Grid: A visual of Open Grid Services Architecture, Forum global sobre Grid. Junio 2003. 86p.

[17] L. Joyanes Aguilar. Cibersociedad. Mac Graw-Hill, Interamericana de España, S.A, Primera edición, Agosto 1997. 1-5p. ISBN: 8448109430.

[18] H.Ñimrod D. Abramson, R. Giddy Sosic. The Grid: Blueprint for a New Computing Infrastructure, IEEE, USA. 1995.

[19] E. Freeman, S. Hupfer, and K. Arnold. JavaSpaces: Principles Patterns, and Practice, Addison-Wesley Publishiers B.V. Enero 1999. Tercera edición, 368p. ISBN: 978-0-201-30955-3

[20] Microsoft Corporation, Información general del framework de .NET Remoting, [en línea], Disponible en WWW: http://msdn.microsoft.com/es-es/library/kwdt6w2k (VS.80).aspx [Consulta: 27 Marzo 2009].

[21] Microsoft Corporation, Arquitectura del framework .NET Remoting, [en línea], Disponible en WWW: http://msdn.microsoft.com/es-es/library/2e7z38xb(VS.80).aspx. [Consulta: 27 Marzo 2009].

[22] Microsoft Corporation, Procesos y Dominios de Aplicación, [en línea], Disponible en WWW: http://msdn.microsoft.com/es-es/library/kt21t9h7(VS.80).aspx. [Consulta: 27 Marzo 2009].

[23] Microsoft Corporation, Periodos y activación de objetos, [en línea], Disponible en WWW: http://msdn.microsoft.com/es-es/library/9ze044wd(VS.80).aspx. [Consulta: 27 Marzo 2009].

[24] Microsoft Corporation, Activación de objetos remotos, [en línea], Disponible en WWW: http://msdn.microsoft.com/es-es/library/cbzcxy2s(VS.80).aspx. [Consulta: 27 Marzo 2009].

[25] Microsoft Corporation, Concesiones del periodo de duración, [en línea], Disponible en WWW: http://msdn.microsoft.com/es-es/library/23bk23zc(VS.80).aspx. [Consulta: 27 Marzo 2009].

[26] Microsoft Corporation, Canales, [en línea], Disponible en WWW: http://msdn.microsoft.com/es-es/library/dkfd3wha(VS.80).aspx. [Consulta: 27 Marzo 2009].

[27] Roger D. Chamberlain & Cheryl Henderson. Evaluating the use of Pre-Simulation in VLSI Circuit Partitioning. Department of Electrical Engineering, Washington University, St. Louis, MO, Julio 1994. Volumen 24. 139-149 p. ISSN: 0163-6103.