PONTIFICIA **UNIVERSIDAD CATÓLICA** DEL PERÚ

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

ADMINISTRADOR DE PROYECTOS DE GRID COMPUTING QUE HACEN USO DE LA CAPACIDAD DE CÓMPUTO OCIOSA DE LABORATORIOS INFORMÁTICOS

Tesis para optar por el Título de Ingeniero Informático, que presenta el bachiller:

Martín Alberto Iberico Hidalgo

ASESOR: Leopoldo Genghis Ríos Kruger

Lima, Junio del 2009

Índice General

Índice de Tablas

Índice de Figuras

RESUMEN

El objetivo de este proyecto es realizar el análisis, diseño e implementación de un administrador de proyectos de grid computing que hacen uso de la capacidad de cómputo ociosa de los laboratorios de la Pontificia Universidad Católica del Perú.

Este documento presenta de manera detallada todo el proceso seguido en el desarrollo de una solución que resuelve el problema planteado de manera eficaz y eficiente. Este proceso incluye la etapa de investigación previa, pasando por el análisis, y diseño del producto final, hasta llegar a la implementación del mismo.

El detalle al que llega este documento, es a mencionar las decisiones de ingeniería tomadas en cada una de las etapas mencionadas, estando cada una de estas, validadas y justificadas, apoyándose en la bibliografía consultada a lo largo de proyecto.

El proceso de aprendizaje, previo al análisis y construcción de la solución propuesta, de las tecnologías investigadas, y aún más, de la tecnología utilizada, fue fundamental en todo el resto del proceso de desarrollo del proyecto.

Finalmente, se muestra el caso de aplicación, basado en la validación de modelos estadísticos de inferencia bayesiana utilizando simulaciones intensivas.

Estas simulaciones necesitan una gran capacidad de procesamiento, debido a que utiliza complejas estructuras matemáticas, así como procesos matemáticos muy largos, como generación de datos de entrada, y procesos de simulación iterativos. La cantidad de iteraciones tiende a ser muy grande, con lo que la precisión de toda la simulación es mucho mayor.

INTRODUCCIÓN

La computación en grid tiene sus orígenes a principios de los 80's donde la investigación intensiva en el campo de la algoritmia, la programación y arquitecturas que soportan paralelismo, provocaron que los primeros éxitos en este campo ayuden a otros procesos de investigación en diversas áreas; además de su exitoso uso en la ingeniería y en el campo empresarial. [16]

La computación en grid, en las comunidades académicas y de investigación, se mantuvo centrada en la creación de marcos de trabajo eficientes para difundir sistemas distribuidos de gran rendimiento. Pero, con la explosión de la Internet y el incremento de poder de computadores de escritorio en el mismo periodo, muchos de los esfuerzos se centraron en crear poderosos sistemas de computación distribuida teniendo como objetivo conectar todas las computadoras asociadas a esta gran red. [20]

El trabajo propuesto pretende implementar una solución que aproveche los ciclos de cómputo no utilizados de los computadores de escritorio en laboratorios informáticos, y con ello resolver problemas computacionalmente intensivos, ayudando y promoviendo a investigadores que necesiten gran capacidad de cómputo, para que puedan empezar proyectos de investigación propios, o concluir de manera efectiva los ya empezados por ellos.

Debido a esto, la computación en grid fue la elegida para implementar la solución al problema planteado, apoyándose en la tecnología BOINC[5] para poder realizar el procesamiento en paralelo.

Además, se plantea el desarrollo de una interfaz de usuario, con lo que la complejidad del grid de computadores sea transparente para el investigador, y éste piense que el procesamiento se realiza en un solo computador.

1 Generalidades

En el presente capítulo se desarrollan los aspectos generales del problema planteado en el documento de tesis. A su vez se presentan los conceptos básicos necesarios para la compresión del problema en todos sus aspectos, de tal forma que se pueda elegir la solución más adecuada. Finalmente se presentará el alcance de la solución que resuelve el problema definido.

1.1 Definición del Problema

Con el incremento de recursos con los que se comercializan los computadores, tales como la cantidad de núcleos de procesamiento, memoria y capacidad de almacenamiento [21], muchos de estos recursos son desperdiciados, ya que la cantidad de carga computacional, que generan los usuarios, no siempre aprovecha todo el potencial que tienen los computadores de escritorio.

A menudo, el recurso ocioso en los computadores, en promedio, es del 85% durante el día y del 95% durante la noche [22], además, se presenta un uso de procesador menor al 5%. [21]

En los últimos años, las desktop grids han emergido como un importante método para utilizar recursos ociosos de computadores de escritorio, y además han sido orientadas hacia computación de alto rendimiento, para procesar una gran cantidad de tareas independientes en aplicaciones científicas o empresariales. [24]

Aunque individualmente, los computadores personales son inferiores en muchos aspectos a los supercomputadores, el poder combinado de cientos de computadores personales, unificados en un sistema en grid, representa un recurso computacional importante. [23]

La estrategia consta en aprovechar la disponibilidad de tiempo ocioso de los computadores personales o de escritorio ubicados en los laboratorios informáticos de la Pontificia Universidad Católica del Perú, agrupándolos bajo un sistema en grid. En nuestro caso, además de aprovechar el recurso ocioso desperdiciado, se pretende aprovechar los recursos no utilizados mientras existan alumnos trabajando en cada uno de estos computadores. Orientando el uso de los computadores como soporte para el desarrollo de proyectos científicos dentro de la universidad, que necesiten realizar procesos computacionalmente intensivos.

Dados estos motivos, se propone realizar el análisis y diseño de una desktop grid, la cual hará uso de los recursos computacionales ociosos antes mencionados, y además se implementará un administrador de proyectos científicos que utilicen esta capacidad de cómputo.

Los laboratorios informáticos de la universidad suman en total 13, y están distribuidos según los distintos tipos de computador con los que se cuentan. En cada laboratorio aproximadamente se tienen 40 computadores. La distribución de los mismos es como sigue:

- GenuineIntel Intel Core 2 CPU 6300 1.8 GHz – 120 computadores distribuidos en un total de 3 laboratorios.
- GenuineIntel Intel Core 2 CPU 6400 2.13 Ghz – 212 computadores distribuidos en un total de 6 laboratorios.
- GenuineIntel Intel Core 2 Duo CPU E8200 2.66 Ghz – 135 computaores distribuidos en un total de 4 laboratorios.

Cabe señalar que estos laboratorios se encuentran en funcionamiento 15 horas al día, por lo que la capacidad de cómputo disponible solo podría ser usada cuando las computadoras se encuentren prendidas.

1.1.1 Objetivo General

Realizar el análisis, diseño e implementación de un Administrador de Proyectos de Grid Computing que hacen uso de la Capacidad de Cómputo Ociosa de los Laboratorios Informáticos de la PUCP.

1.1.2 Objetivos Específicos

A. Definir la tecnología a utilizar para hacer el procesamiento en paralelo.
B. Definir la estructura de datos de ingreso a la aplicación que realizará el proceso requerido, si es que los necesitara.
C. Elaborar la aplicación que realizará la ejecución del proceso de computación intensivo.
D. Definir y acoplar la estructura de Base de Datos a utilizar en el Administrador de Proyectos.
E. Diseñar e implementar el Administrador de Proyectos.
F. Definir y realizar las validaciones correspondientes de los datos de salida generados.
G. Diseñar e implementar el monitoreo interno en la Base de Datos de las tareas en ejecución.
H. Diseñar e implementar el procesamiento de los datos de salida.
I. Diseñar e implementar las pruebas de rendimiento y esfuerzo al sistema desarrollado.

1.1.3 Resultados Esperados

A. Documento de Visión.
B. Documento de Análisis.
C. Especificación de Requisitos de Software.
D. Código fuente del algoritmo que realice el procesamiento en paralelo.
E. Código fuente del generador de tareas.
F. Código fuente del algoritmo que realice el procesamiento de salida.
G. Diagrama entidad-relación de la base de datos.
H. Prototipos de Interfaz Gráfica.

1.2 Marco Conceptual

Se describen los conceptos básicos necesarios para la compresión del problema en todos sus aspectos, para posteriormente plantear una solución satisfactoria y eficiente. Primero se presentan los conceptos básicos del flujo de la solución, para luego presentar los aspectos computacionales, los cuales son los puntos más estudiados y profundizados en el documento.

1.2.1 Computación Paralela

Por muchos años la computación ha sido sinónimo de programación secuencial, donde la idea de un algoritmo está basada en la secuencia de instrucciones. La programación no secuencial se transforma en un tópico de investigación, a medida que investigadores reconocen que los sistemas operativos pueden ser rediseñados y construidos bajo la premisa de ser una colección de procesadores independientes trabajando en forma colaborativa. Con esta idea nace el concepto de computación concurrente, interacciones entre hilos de trabajo independientes. [15]

Numerosos paradigmas han sido desarrollados para la computación distribuida y paralela. Además, diferentes ambientes de desarrollo e infraestructuras enfocadas a estos paradigmas están disponibles. [18]

La computación paralela consiste en agrupar computadores independientes de forma que la imagen de éstos hacia el usuario final sea la de un único computador cuyas propiedades, en la medida de lo posible, sea la suma de las prestaciones de los equipos que formen el sistema paralelo. [18]

Esta agrupación de computadores tiene como objetivo realizar una tarea o proceso computacionalmente intensivo, y para ello, la tarea necesita ser descompuesta en distintas subtareas independientes, debiendo redireccionar cada una de éstas hacia un computador asociado al proyecto, el cual tenga los recursos necesarios para la resolución de las mismas. A medida que las subtareas sean completadas por cada uno de los computadores, se centralizan en uno de ellos para formar la tarea inicial, obteniendo el resultado en un tiempo mucho menor si es que se realizara en un sólo computador.

Así, como al resolver un problema computacional con un solo computador, en el cual necesitamos una secuencia ordenada de pasos para la resolución del problema, al cual conocemos como algoritmo; en la resolución de un problema utilizando varios computadores, necesitamos también un algoritmo con el cual se resuelve el problema, pero además necesitamos especificar los pasos que pueden ser ejecutados simultáneamente, donde cada uno de ellos debe ser independiente del resto. Esto es esencial para obtener un buen rendimiento a la hora de ejecutar el algoritmo en forma paralela. [4]

Al momento de especificar un algoritmo paralelo debemos tener en cuenta los siguientes pasos a realizar [4]:

- Identificar las porciones de código que puede ser ejecutada en paralelo.
- Mapear las porciones de trabajo que se ejecutan en paralelo en los diferentes computadores pertenecientes al grid.
- Distribuir los datos de ingreso, las salidas y la información intermedia en la aplicación.
- Controlar el acceso a la información compartida requerida por el programa por los diferentes computadores.
- Sincronizar los diferentes computadores teniendo en cuenta el algoritmo paralelo.

La computación paralela ha tenido un impacto tremendo en una gran cantidad de áreas, como la científica, en aplicaciones de ingeniería, aplicaciones comerciales, así como simulaciones computacionales.

1.2.2 Computación en Grid

La computación en grid tiene sus orígenes a principios de los 80's donde la investigación intensiva en el campo de la algoritmia, la programación y arquitecturas que soporten paralelismo, provocaron que los primeros éxitos en este campo, ayuden a otros procesos de investigación en diversas áreas; además de su exitoso uso en la ingeniería y en el campo empresarial. [16] Hoy en día, los entornos de computación en malla son las más prometedoras infraestructuras en la investigación computacional. [17]

Una grid computacional es la infraestructura de hardware y software que provee acceso fiable, consistente y a bajo costo, a altas capacidades computacionales. [17] Estas capacidades computacionales son explotadas mediante el desarrollo de aplicaciones diseñadas bajo la perspectiva de la programación paralela o distribuida, para la optimización de procesos que aletarguen el flujo que se desea optimizar. Estos procesos deben cumplir la precondición de poderse subdividir en procesos más pequeños, unos algorítmicamente independientes de los otros.

Una característica particular de la computación en grid, es que los nodos o equipos que forman el grid no tienen por qué ser computadores dedicados. Esta propiedad, denominada pertenencia dinámica, implica que cualquier equipo puede adherirse o abandonar un determinado grid en tiempo de ejecución. Además, el hecho de que un equipo este formando, en un momento dado, parte de un determinado grid, no implica que deje de ser operativo para cualquier otro tipo de tarea, sino, que en diferentes instantes de tiempo, un porcentaje de sus recursos, tales como el uso de CPU o almacenamiento secundario, serán utilizados por el grid para la ejecución de determinadas tareas. [18]

1.2.3 Desktop Grid

La donación de ciclos de procesamiento inutilizados ha sido una idea bastante aceptada entre las personas alrededor del mundo, quienes se han unido voluntariamente a proyectos de investigación en distintas áreas. Todos estos proyectos están basados en la idea de enrolamiento de personas, siendo éstas últimas las que toman la decisión de brindar ciclos computacionales de

sus computadores personales para ser utilizados por un proyecto de investigación, elegido según sus intereses.

Luego de la suscripción a un proyecto específico, un pequeño programa de control es descargado al computador. Este programa es responsable de la comunicación con el servidor central del proyecto, así como el uso de la capacidad brindada, ejecutando procesos enviados por este servidor. Típicamente, estos proyectos utilizan pequeños paquetes de comunicación para manejar grandes procesamientos en el computador asociado al proyecto. Esto es con la intención de ser lo menos intrusivos en el computador y en su conexión a Internet. [20]

Desktop Grid es una creciente tendencia en la computación en grid. En contraparte a los sistemas en grid tradicionales, en desktop grids los operadores de los sistemas en grid brindan las aplicaciones y los usuarios de los mismos proveen los recursos para procesar estas aplicaciones. [25]

Las Desktop Grids presentan las siguientes características:

- Presentan un conjunto de computadores unidos a una red compartida. Este conjunto puede poseer computadores dedicados, computadores conectados de forma intermitente y computadores compartidos con otras actividades.
- Presentan un conjunto de políticas describiendo la forma en como cada uno de los computadores participan en el grid.
- Cada computador desconoce de la existencia de los demás computadores, exceptuando el servidor central del proyecto.
- Un mecanismo de control de envío, ejecución y recuperación de las tareas a procesar, todo esto bajo el control del servidor central del proyecto.

1.2.4 BOINC [5]

BOINC, por sus siglas en inglés, Infraestructura Abierta de Berkeley para la Computación en Red, es una plataforma de código abierto que permite a proyectos hacer uso de la capacidad de cómputo de diversos computadores,

ubicados alrededor del mundo, de manera voluntaria. Es decir, que cuando un computador conectado a la red BOINC se encuentra inactivo, inmediatamente empieza a trabajar para el proyecto al cual se ha unido, procesando los datos que le solicita al proyecto al cual está unido. Estos proyectos usualmente están relacionados a investigaciones científicas de diversos tipos.

Además, esta plataforma puede utilizarse para realizar computación en red dentro de una misma organización, por ejemplo para realizar cálculos propios que la misma empresa necesita dentro de su negocio.

Las características más importantes de BOINC se muestran a continuación:

- Autonomía de proyectos. Varios proyectos pueden utilizar BOINC para realizar el procesamiento. Cada uno de los cuales se maneja como un proyecto independiente, el cual posee su propio servidor y base de datos.
- Flexibilidad del voluntariado. Cada uno de los voluntarios pueden participar en múltiples proyectos, dependiendo de los intereses de cada uno, y además, decidir la cantidad de recursos donados a cada uno de los proyectos en los cuales participa.
- Multiplataforma. El BOINC cliente es encontrado para múltiples plataformas, como Windows, Linux y MacOS.

BOINC ha sido diseñado para soportar aplicaciones que necesiten grandes cantidades de recursos de cómputo, siendo el único requerimiento que esta aplicación sea divisible en gran cantidad de tareas pequeñas que puedan ser realizadas, cada una, de manera independiente.

1.2.5 Interfaz de Programación de Aplicaciones - API

BOINC posee una interfaz de programación desarrollada íntegramente en C++, y muchas de las cuales poseen interfaces en C, por lo que pueden ser utilizadas en programas escritos en éste y otros lenguajes.

Las funciones que conforman el API de BOINC se dividen en cuatro tipos, las funciones básicas, las funciones gráficas, las funciones de manejo de mensajes y las funciones de envío de archivos.

Las funciones básicas están encargadas de la inicialización y finalización de aplicaciones, resolución de nombres de archivos, control de progreso de unidades de trabajo, manejo de secciones críticas, obtención y reporte de creditaje (utilizado en la computación voluntaria) y la comunicación con el cliente. Todo ello es utilizado para monitorear el proceso de cálculo que en realidad se está efectuando dentro de cada uno de los computadores cliente.

Una aplicación BOINC también puede producir gráficos mediante las funciones gráficas del API. Esto se utiliza, en los proyectos de computación voluntaria, como SETI@Home[6], del cual hablaremos posteriormente, para mostrar un salvapantalla con cierta información de los datos procesados, con la finalidad de darle un aspecto visual agradable al proyecto.

El envío de mensajes permite, a una aplicación, comunicarse con el servidor en el proceso de ejecución de una unidad de trabajo. Esta comunicación se realiza mediante archivos XML.

Una aplicación debe tener un archivo como resultado de su proceso de cálculo, el cual debe ser enviado hacia el servidor, por lo que se necesita una serie de funciones que hagan el manejo de los resultados parciales.

1.2.6 Aplicaciones Heredadas

Las aplicaciones heredadas son aplicaciones que no han utilizado el API de BOINC para su desarrollo. Dentro de este conjunto de aplicaciones tenemos a aquellas que no fueron desarrolladas para ejecutarse bajo BOINC o aquellas de las cuales no poseemos el código fuente, sino que sólo poseemos el ejecutable. Estas aplicaciones son manejadas por otra aplicación que si ha utilizado el API de BOINC en su desarrollo, llamada “wrapper”, la cual ha sido desarrollada íntegramente en el lenguaje C. Ésta se encarga de la comunicación con el cliente y ejecuta la aplicación heredada como un subproceso. La figura 1.1 ilustra este punto.

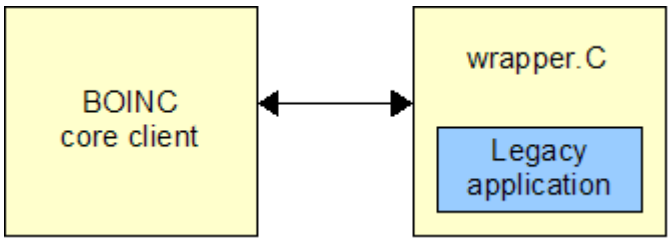

Figura 1.1 - Legacy Applications [5]

El programa wrapper lee un archivo XML de donde obtiene toda la información necesaria para la ejecución del programa heredado, como los parámetros de entrada, de salida y el manejo de errores. El nombre de la aplicación heredada y los argumentos por la línea de comandos también se leen de este archivo. Los programas heredados deben devolver un número entero, cuando este número es cero, el programa wrapper lo toma como un éxito, en caso contrario, se toma como un fracaso.

1.3 Estado del Arte

En la actualidad existen diferentes proyectos que resuelven problemas utilizando la grid computing, por lo que se desarrollarán aquellos que específicamente realizan esta clase de computación.

1.3.1 Proyectos BOINC

Como se ha mencionado en el punto 1.2.4, BOINC es un software intermedio que conecta diversos servicios que hacen posible la computación voluntaria. Se han desarrollado diversos proyectos que utilizan esta tecnología para poder captar capacidad de procesamiento ociosa en el mundo entero y poder procesar la gran cantidad de información que cada uno de estos proyectos posee. Entre los proyectos más importantes tenemos:

SETI@Home [6]

Por sus siglas en inglés, Búsqueda de Inteligencia Extraterrestre, es el proyecto por el cual BOINC fue creado, para salvaguardar algunos aspectos de

seguridad y de control de los resultados que eran enviados por los usuarios pertenecientes a la red.

SETI es una red científica liderada por el David Anderson, que también lidera el equipo encargado de BOINC, cuya meta es la detección de vida inteligente fuera de nuestro planeta. El proyecto posee radio telescopios que monitorean el espacio captando ondas de radio de banda estrecha, las cuales se saben, no ocurren de forma natural, por lo que su detección sería una evidencia de tecnología extraterrestre.

Las señales de los radiotelescopios consisten en ruidos provenientes de fuentes celestiales y señales producidas por el hombre. Los proyectos de radio SETI se encargan del análisis de estas señales, las cuales se procesan digitalmente, por lo que una potencia mayor de cálculo permite que la búsqueda cubra un mayor rango de frecuencias con una mayor sensibilidad.

Inicialmente estos cálculos se realizaban mediante computadoras en los mismos radiotelescopios, las cuales se encargaban de procesar la mayor cantidad de información. En 1995, David Geyde, propuso que radio SETI tuviera una supercomputadora virtual conectada a través de Internet. El proyecto SETI@Home fue lanzado en mayo de 1999.

El proyecto SETI@Home posee diversos recursos computacionales, como servidores de base de datos para información de usuarios, listas, unidades de trabajo, resultados, información procesada; servidores web para el hosting de la web del proyecto. Cada uno de estos servicios está dividido en 10 servidores, la mayoría de los cuales utilizan tecnología Intel.

Entre los principales patrocinadores del proyecto se encuentran:

- The Planetary Society
- Sun Microsystems
- Intel
- ProCurve Networking
- SnapAppliance
- Quantum

- XILINX

EINSTEIN@Home [7]

Al igual que SETI@Home, EINSTEIN@Home es un proyecto de computación voluntaria que analiza ondas gravitacionales producidas por fuentes de ondas continuas, las cuales pueden incluir púlsares. El nombre proviene del científico alemán, Albert Einstein, quien a principios del siglo XX predijo la ocurrencia de estas ondas gravitacionales.

El proyecto fue lanzado oficialmente el 19 de Febrero del 2005 como contribución por parte de la Sociedad Americana de Física para el World Year of Physics 2005.

El objetivo científico del proyecto es la búsqueda de fuentes de radiación gravitacional de onda continua. El éxito en la detección de ondas gravitacionales constituiría un hito importante en la física, ya que nunca antes se ha detectado un objeto astronómico únicamente por radiación gravitacional.

La información es obtenida mediante dos fuentes, por el Laser Interferometer Gravitational-Wave Observatory, y mediante GEO 600, otro detector de ondas gravitacionales.

Rosetta@Home [8]

Rosetta@Home es un proyecto orientado a determinar las formas tridimensionales de las proteínas a través de investigaciones científicas experimentales que a la larga podrían llevar a descubrir curas para las más grandes enfermedades humanas, como el VIH, la malaria y el cáncer.

Todos los proyectos antes mencionados son parte de la computación voluntaria, donde usuarios al rededor del mundo donan su tiempo de cómputo desperdiciado a diversos proyectos, los cuales pueden ser elegidos, por ejemplo, por interés en el tipo de proyecto planteado.

Otros proyectos

Muchos otros proyectos, en distintas áreas de investigación, utilizan la tecnología BOINC, donde cabe destacar los siguientes:

- Matemáticas, Juegos de Estrategia, 3D, Informática
  - SZTAKI DesktopGrid
  - Chess960@Home
  - Rectilinear Crossing Number
  - Riesel Sieve
  - VTU@Home
  - Render Farm
  - Prime Grid
  - Xtrenelab
  - ABC@Home
  - DepSpi
- Biología y Medicina
  - Malariacontrol.net
  - Predictor@Home
  - World Community Grid
  - SIMAP
  - Tanpaku
  - Ralph@Home
  - Docking@Home
  - Project Neuron
  - Proteins@Home
- Astronomía, Física y Química
  - QMC@Home
  - LHC@Home
  - Spinhenge@Home
  - Leiden Classical
  - Hash Clash
  - Ufluids
  - Orbit@Home
- Ciencias de la Tierra

- Climateprediction
- BBC Climate Change Experiment
- Seasonal Attribution Project

1.3.2 Terracota

Terracota es una infraestructura de software open source que permite ejecutar una aplicación hecha en Java en cuantos computadores sean necesarios para tal fin, sin tener que elaborar un código específico para poder ser ejecutado en un cluster convencional. [26]

La arquitectura de Terracota tiene dos elementos principales, los nodos cliente y el Terracota Server Array.

Cada nodo cliente corresponde a un proceso Java dentro del cluster. Estos nodos ejecutan una Máquina Virtual Java estándar, logrando cargar a Terracota mediante sus librerías; todo esto ocurre cuando la Máquina Virtual Java se inicializa.

El Terracota Server Array provee una inteligencia de clustering tolerante a fallos, además de instalaciones de alto rendimiento. Cada instancia del Servidor Terracota dentro del array es 100% procesos Java.

Terracota utiliza la tecnología Network-Attached Memory (NAM) para permitir realizar el clustering con Máquinas Virtuales Java. [26] Como se muestra en la imagen 1.2, Terracota es la Máquina Virtual Java para la aplicación a ejecutar dentro del cluster, mientras que para la verdadera Máquina Virtual Java, Terracota es la aplicación. [27]

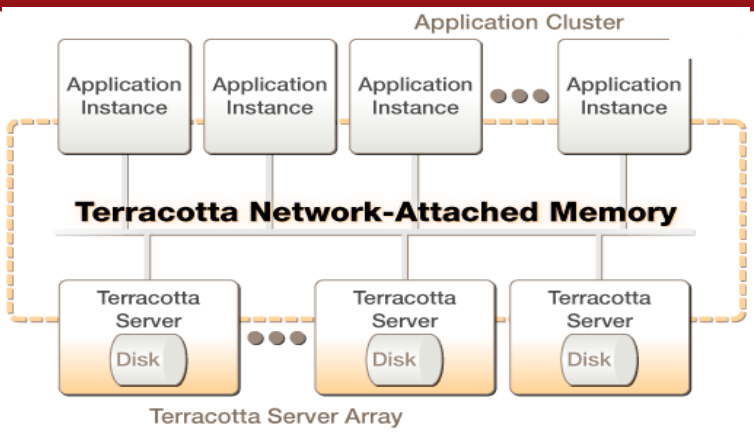

Figura 1.2 La Arquitectura Terracota [26]

Con Terracota, un cambio en una Máquina Virtual Java se ve reflejado casi instantáneamente en las demás máquinas virtuales que necesitan saber del cambio producido.

1.3.3 The Globus Toolkit

The Globus Toolkit es un conjunto de módulos independientes [28] de código abierto elaborados para el desarrollo de servicios orientados a las aplicaciones de computación distribuida [29], permitiendo a personas compartir poder computacional, bases de datos, y otras herramientas, de forma segura a través de los límites de corporaciones e instituciones, sin sacrificar autonomía geográfica [30].

El toolkit se basa en tecnologías estándar como XML[A], SOAP[B], WSDL[C], Web Services Resource y ha sido implementado íntegramente en Java.

Globus Toolkit intenta proporcionar un sistema de componentes estándar que pueda dar soporte a una gran variedad de aplicaciones personalizadas sin la necesidad de desarrollar completamente una infraestructura a medida para cada aplicación. No proporciona una solución “lista para usar”, sino que proporciona bloques constructivos y herramientas estándares para ser usados por los desarrolladores e integradores del sistema. [31]

1.3.4 Sun Grid Engine [32]

El Sun Grid Engine es un esfuerzo de la comunidad open source para la adopción de soluciones de computación distribuida. El proyecto Grid Engine proporciona software para la gestión de recursos distribuidos para grandes necesidades de potencia computacional.

El proyecto nace a partir de la compra, por parte de Sun Microsystems, de Gridware, un desarrollo privativo de Gestión de Recursos Distribuidos, DRM por sus siglas en ingles.

Dentro de las misiones que han sido tomadas como horizonte dentro del proyecto, se tiene:

- Brindar tecnología de punta en Grid Engine.
- Establecer estándares para guiar a la comunidad en computación distribuida.
- Permitir que desarrolladores puedan crear innovadores servicios y aplicaciones a través de ambientes computacionales distribuidos y heterogéneos.

El software Transfer-queue Over Globus (TOG) permite a las empresas acceder a recursos computacionales a través de su existente instalación de Grid Engine. TOG integra Grid Engine 5.3 y el Globus Toolkit 2.2.4, visto en el punto 1.3.3.

El Globus Toolkit es esencialmente una API que permite interconectar recursos computacionales. La integración con Globus, permite a Grid Engine, soportar la colaboración entre empresas, permitiendo a cada una de ellas acceder a recursos remotos que cada una posee.

TOG ha sido utilizado para crear proyectos de computación en grid entre las universidades de Glasgow y Edimburgo, en el Reino Unido. Investigadores en la Universidad de Glasgow, lugar del Centro Nacional de e-Ciencia, han sido capaces de acceder a recursos computacionales de la EPCC utilizando una configuración hecha con Grid Engine y TOG.

1.4 Descripción y Sustentación de la Solución

Como ya se ha visto, la capacidad de cómputo ociosa en los computadores de escritorio es bastante grande en comparación con la cantidad de recursos que realmente se aprovechan. Esta capacidad puede canalizarse hacia proyectos de investigación computacionalmente intensivos, los cuales no pueden realizarse en computadores de escritorio debido a limitaciones principalmente de hardware, aprovechando así los recursos con que la universidad cuenta, promoviendo la investigación dentro del campus.

La solución del problema presentará el diseño de un grid de computadores, todas trabajando bajo la tecnología BOINC, la cual estará sobre un servidor Linux; mientras que cada uno de los nodos pertenecientes al grid, manejarán el sistema operativo Windows XP. Existe además, la posibilidad de implementar la solución con computadores cliente que tengan instalado un sistema operativo Linux. El grid computacional será implementado para aprovechar los recursos computacionales ociosos en los laboratorios informáticos de la universidad.

1.5 Planificación

La efectiva administración de un proyecto de software depende en un profundo planeamiento del progreso del proyecto. Un esquema previo del proyecto debe utilizarse como un manejador del mismo.

1.5.1 Descomposición de Tareas

Se presenta una visión general de las etapas que se pretenden llevar a cabo durante la concepción del proyecto, con una estimación de 779 horas de trabajo.

1.5.2 Etapa de Concepción

En esta etapa se pretender alcanzar un conocimiento completo sobre el problema a resolver, teniendo como base la investigación preeliminar realizada.

El nivel teórico que se necesita incluye el conocimiento de la computación en grid, y el conocimiento de herramientas que ayudarán a que la implementación de la solución sea la óptima. Todo este ha sido visto en los puntos 1.1, 1.2 y 1.3.

El estudio de las técnicas de computación paralela, la forma de comunicación entre computadores cliente hacia un computador principal, y viceversa, además de técnicas de descomposición de algoritmos, son necesarias, y al final de esta etapa se tendrá una idea clara de cómo funcionará el algoritmo que trabaje de forma eficiente en varios computadores de forma paralela.

1.5.3 Etapa de Elaboración

En esta etapa se consideran los aspectos básicos y fundamentales de la investigación, además de la elaboración de documentación útil para los siguientes pasos del proyecto. Se documentan las técnicas estudiadas en la etapa anterior, y se escogen las que se adecuan a resolver el problema de forma eficiente, teniendo como objetivo final la elaboración de documentos que marquen los siguientes pasos a seguir. El método de descomposición del algoritmo para que trabaje en paralelo será elegido, el cual se justificará de forma apropiada. La arquitectura general del grid de computadores a diseñar, además de la arquitectura del sistema administrador de proyectos son definidos en esta etapa del desarrollo.

1.5.4 Etapa de Construcción

En esta etapa se realiza la implementación del grid computacional, anexando todos los computadores disponibles al servidor central. Además, se desarrollará el sistema administrador de proyectos que hará uso del grid. Se harán pruebas con algoritmos de fácil manejo en principio, y luego se escalará en niveles de dificultad y procesamiento para constatar que funcione correctamente, no se pierda información y para que la carga de procesamiento sea la adecuada en cada uno de los nodos. Para ello, se utilizarán herramientas propias del sistema operativo. En la etapa final se juntarán ambos

núcleos de la solución, el sistema administrador de proyectos y el grid de computadores para poder realizar pruebas con el sistema integrado y a mayor escala, esperando obtener resultados adecuados.

1.5.5 Etapa de Transición

Esta es la etapa final del proyecto, en la que se tendrá el producto terminado. Luego de las últimas pruebas a realizarse, se tendrá la versión candidata del producto, para luego hacer las mejoras del caso y tener la versión final. Posteriormente se empezarán a realizar las mediciones necesarias para demostrar que la implementación final cumple las expectativas iniciales y resuelve en su totalidad el problema planteado inicialmente.

La siguiente figura muestra todas las etapas de concepción del proyecto, como los tiempos de duración estimados para cada una de ellas.

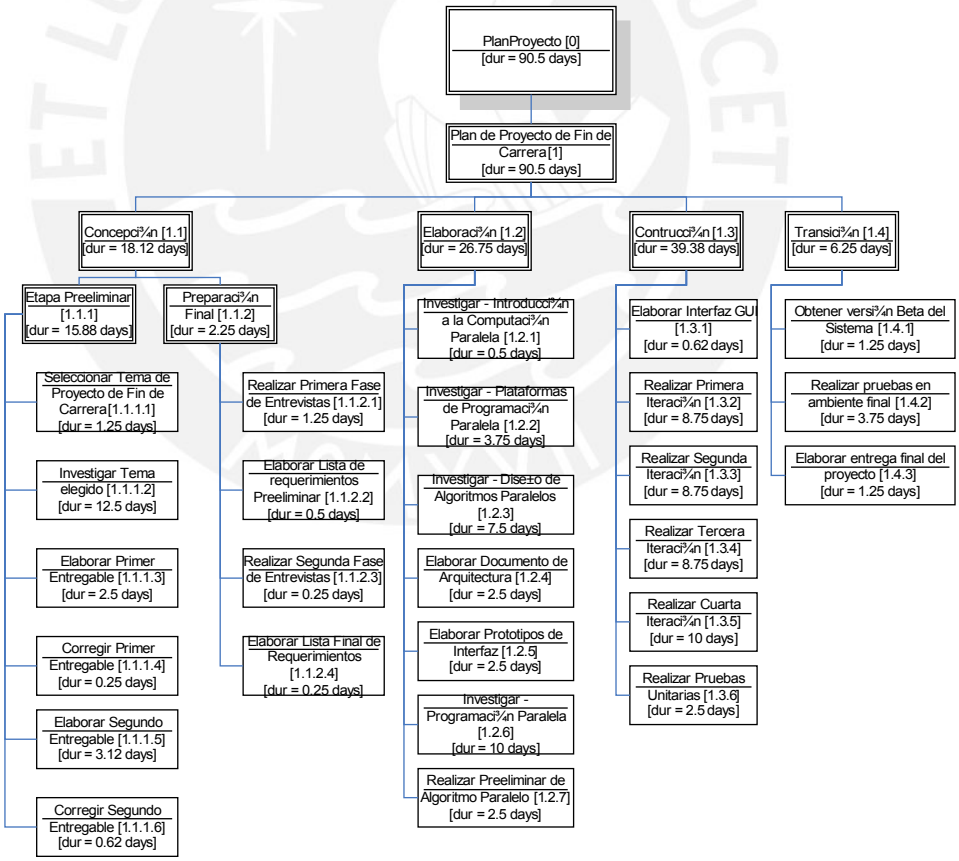

Figura 1.3 - Work Breakdown structure - WBS

1.6 Diagrama de Gantt

El diagrama de Gantt muestra de manera detalla la estimación realizada con respecto a los tiempos para la culminación del proyecto. El diagrama de Gantt se encuentra resumido en la siguiente tabla y en la siguiente figura.

	Horas Estimadas	Fecha Inicio	Fecha Fin
Fase de Concepción	180	30/08/07	28/10/07
Fase de Elaboración	234	28/10/07	24/01/08
Fase de Construcción	315	24/01/08	02/06/08
Fase de Transición	50	02/06/08	22/06/08

Tabla 1.1 – Diagrama de Gantt

	Nombre de tarea	Duración	WBS	Comienzo	Fin	Predecesoras
1	**- Plan de Proyecto de Fin de Carrera**	**90.5 días**	**1**	**jue 30/08/07**	**dom 22/06/08**	
2	**- Concepción**	**18.13 días**	**1.1**	**jue 30/08/07**	**dom 28/10/07**	
3	**- Etapa Preeliminar**	**15.88 días**	**1.1.1**	**jue 30/08/07**	**sáb 20/10/07**	
4	Seleccionar Tema de Proyecto de Fin de Carrera	10 horas	1.1.1.1	jue 30/08/07	dom 02/09/07	
5	Investigar Tema elegido	100 horas	1.1.1.2	lun 03/09/07	sáb 13/10/07	4
6	Elaborar Primer Entregable	20 horas	1.1.1.3	lun 03/09/07	lun 10/09/07	4
7	Corregir Primer Entregable	2 horas	1.1.1.4	vie 28/09/07	sáb 29/09/07	6
8	Elaborar Segundo Entregable	25 horas	1.1.1.5	sáb 29/09/07	lun 08/10/07	7
9	Corregir Segundo Entregable	5 horas	1.1.1.6	vie 19/10/07	sáb 20/10/07	8
10	**- Preparación Final**	**2.25 días**	**1.1.2**	**dom 21/10/07**	**dom 28/10/07**	**3**
11	Realizar Primera Fase de Entrevistas	10 horas	1.1.2.1	dom 21/10/07	jue 25/10/07	
12	Elaborar Lista de requerimientos Preeliminar	4 horas	1.1.2.2	jue 25/10/07	sáb 27/10/07	11
13	Realizar Segunda Fase de Entrevistas	2 horas	1.1.2.3	sáb 27/10/07	sáb 27/10/07	12
14	Elaborar Lista Final de Requerimientos	2 horas	1.1.2.4	sáb 27/10/07	dom 28/10/07	13
15	**- Elaboración**	**26.75 días**	**1.2**	**dom 28/10/07**	**jue 24/01/08**	**2**
16	Investigar - Introducción a la Computación Paralela	4 horas	1.2.1	dom 28/10/07	lun 29/10/07	10
17	Investigar - Plataformas de Programación Paralela	30 horas	1.2.2	mar 30/10/07	dom 11/11/07	16
18	Investigar - Diseño de Algoritmos Paralelos	60 horas	1.2.3	dom 11/11/07	jue 06/12/07	17
19	Elaborar Documento de Arquitectura	20 horas	1.2.4	jue 06/12/07	vie 14/12/07	18
20	Elaborar Prototipos de Interfaz	20 horas	1.2.5	jue 06/12/07	vie 14/12/07	18
21	Investigar - Programación Paralela	80 horas	1.2.6	vie 14/12/07	mié 16/01/08	20
22	Realizar Preeliminar de Algoritmo Paralelo	20 horas	1.2.7	mié 16/01/08	jue 24/01/08	21
23	**- Contrucción**	**39.38 días**	**1.3**	**jue 24/01/08**	**lun 02/06/08**	**15**
24	Elaborar Interfaz GUI	5 horas	1.3.1	jue 24/01/08	sáb 26/01/08	
25	Realizar Primera Iteración	70 horas	1.3.2	sáb 26/01/08	dom 24/02/08	24
26	Realizar Segunda Iteración	70 horas	1.3.3	dom 24/02/08	lun 24/03/08	25
27	Realizar Tercera Iteración	70 horas	1.3.4	lun 24/03/08	lun 21/04/08	26
28	Realizar Cuarta Iteración	80 horas	1.3.5	mar 22/04/08	sáb 24/05/08	27
29	Realizar Pruebas Unitarias	20 horas	1.3.6	dom 25/05/08	lun 02/06/08	28
30	**- Transición**	**6.25 días**	**1.4**	**lun 02/06/08**	**dom 22/06/08**	**23**
31	Obtener versión Beta del Sistema	10 horas	1.4.1	lun 02/06/08	vie 06/06/08	
32	Realizar pruebas en ambiente final	30 horas	1.4.2	vie 06/06/08	jue 19/06/08	31
33	Elaborar entrega final del proyecto	10 horas	1.4.3	jue 19/06/08	dom 22/06/08	32

Figura 1.4 – Diagrama de Gantt

2 Análisis

En el presente capítulo se presenta la etapa de análisis, en la cual se basará todo el proceso de desarrollo del software. Esto incluye la elección del método a utilizar en todo el proceso, la identificación de los requerimientos, y el análisis de la solución a implementar.

2.1 Definición de la metodología de solución

Desde los inicios de la computación en grid, muchas plataformas fueron desarrolladas para realizar este tipo de tareas, plataformas de envío de mensajes fueron la norma en estos tiempos, pasados los años, modelos de programación para estas plataformas fueron desarrollados, las cuales fueron

agrupadas en API's, como PVM[D] y MPI[E], librerías como POSIX[F] fueron aceptadas como estándar. [4]

Para poder tener conectividad entre el servidor central y los computadores anexados al grid, se necesita una interfaz con la que la administración del servidor central, como la de los computadores asociados, se maneje de manera automática, pudiendo centrar el desarrollo en una aplicación intermedia, entre el cliente y la interfaz necesitada.

La interfaz elegida es BOINC, teniendo como referencia la gran cantidad de proyectos científicos los cuales la utilizan para realizar computación voluntaria, los cuales se mencionan en el Marco Conceptual, teniendo la posibilidad de realizar computación en grid en el ambiente en el que se genera el problema. Además de poseer una interfaz web donde se pueden monitorear todos los computadores asociados al grid, todas las unidades de trabajo que han sido enviadas y en que computador están siendo ejecutadas, pudiendo conocer además el progreso de las mismas.

El punto inicial para la construcción de la solución, se basa en el conocimiento sólido de la tecnología que se utilizará en el desarrollo de la misma, en este caso BOINC, como administrador y cliente del grid de computadores.

Se toma la decisión de dividir el proyecto en tres etapas, dadas las características tan particulares de cada una, y su nivel de prioridad en el desarrollo de la solución. A continuación se presentan las etapas definidas para desarrollar el proyecto, en función a su nivel de importancia para el desarrollo de la solución en su totalidad.

- Desarrollo de la aplicación cliente
- Desarrollo de la interfaz Legión
- Desarrollo de la aplicación de procesamiento de salida

Como ya se dijo, cada una de las etapas en las que se dividió el proyecto, tienen características muy particulares, por lo que la elección de una sola metodología, que englobe toda la solución, es inadecuada y además

ineficiente, ya que muchas de estas metodologías están orientadas a proyectos de mayor envergadura, y la utilización de muchos de sus componentes son irrelevantes para el desarrollo de este proyecto. Dado esto, se ha decidido que para cada una de estas etapas se utilice una adaptación de las metodologías conocidas, las cuales se explicarán en el siguiente punto del documento.

2.1.1 Desarrollo de la aplicación Cliente

Esta aplicación forma parte de la primera etapa del desarrollo, donde la exploración de la nueva tecnología utilizada, BOINC, en su adaptación para la satisfactoria solución, es crítica para el desarrollo global del producto. La aplicación cliente es controlada en su totalidad por la aplicación administrativa del BOINC, manteniendo la independencia en cuanto a la manipulación de la información de entrada, dado que el desarrollo en esta etapa se realiza independientemente de la tecnología, pero hay que tener en cuenta que la administración corre por parte del BOINC.

En la primera etapa del desarrollo se utiliza un enfoque de la metodología de desarrollo Xtreme Programming, XP. Dado que no se tenía de forma clara hacia donde debía apuntar el desarrollo de esta etapa, ya que la nueva tecnología se encontraba aun en estudio, y a medida que se avanzaba en el análisis y en el desarrollo mismo, aparecían nuevos requerimientos y nuevas formas de cómo resolver ciertos problemas puntuales. Esto hace factible que las iteraciones, en las que se subdividió esta parte del desarrollo, sean muy cortas y con objetivos pequeños y muy puntuales. Con esto notamos que los requerimientos fueron bastante variables a lo largo del desarrollo. Dadas las características mencionadas, y ya que la metodología se adapta a éstas, su elección queda fundamentada. [9]

No se puede aplicar toda la metodología XP, ni cualquier otra metodología como RUP o adaptaciones de ésta, ya que en esta etapa, el cliente es el mismo desarrollador, dado que él mismo da cuenta de lo necesario a medida que va creciendo el desarrollo.

El proceso de pruebas es definido en base a los pequeños objetivos planteados, y a los resultados esperados obtenidos en el análisis, y como se ha mencionado antes, el descubrimiento de nuevas posibilidades de solución planteados por la herramienta, hace posible que diferentes formas de resolver el problema sea posible, por lo que las implementaciones y las pruebas necesitadas sean aun más rigurosas para cada una de las posibilidades. Este proceso de pruebas no es ejecutada por software, sino que se basa en un proceso de prueba y error ejecutado por el mismo desarrollador, ya que la complejidad de implementación de los requerimientos hasta este punto no son tan complejos de desarrollar. Cabe resaltar que todo el conocimiento previo obtenido hace posible que, lo afirmado en el punto anterior, sea válido.

2.1.2 Desarrollo de la Interfaz Legión

El punto de inicio para el desarrollo de la interfaz, es necesario automatizar la generación de tareas en base a los parámetros que serán ingresados por el investigador mediante la interfaz web. El generador de tareas será el encargado de la generación de unidades de trabajo dentro del esquema con el que BOINC trabaja. Dado que cada una de las unidades de trabajo debe procesar una parte de la tarea que se requiere, dentro del generador debe encontrarse la lógica con la cual se diferencian cada una de las unidades, y con lo que se define la parte del procesamiento que le corresponde.

Para el desarrollo de la interfaz Legión es conveniente utilizar una adaptación de la metodología Rational Unified Process, RUP [10]. La interfaz depende exclusivamente de las necesidades del cliente final, como no lo es así en la etapa anteriormente descrita, por lo que una administración de requerimientos adecuada es fundamental, y se necesita una visión global de los alcances de todo el desarrollo para validar la viabilidad del mismo. Esta parte se encuentra estrechamente ligada al desarrollo de toda la solución en su conjunto, ya que el resto del desarrollo será invisible para el usuario final.

En el plan de proyecto, elaborado en esta etapa, se toma en cuenta todo el desarrollo que se llevará acabo de aquí en adelante, tomando el desarrollo anterior totalmente transparente, ya que la interfaz no depende en lo absoluto del primer punto desarrollado, sino que está estrechamente ligada al proceso inicial del BOINC, específicamente la creación de las tareas a realizar; mientras

que el BOINC administra la aplicación desarrollada en el primer punto, y se vincula a cada una de las máquinas pertenecientes al grid. [11]

El catálogo de Requisitos y los diagramas de casos de uso planteados por RUP, se consideran una buena forma de captar y administrar los requerimientos, y además sirve de guía para el posterior diseño, previo análisis de los requerimientos mencionados anteriormente.

Todo el proceso de construcción es realizado en cuatro iteraciones. En la primera se pretende desarrollar el proceso fundamental del sistema, la generación de tareas vía sistema; en la segunda iteración se pretende desarrollar el seguimiento del proceso de ejecución de las mismas y la descarga de resultados; en la tercera etapa se pretende desarrollar la administración de usuarios y la seguridad del sistema; mientras que en la etapa final se desean desarrollar algunos procesos secundarios muy pequeños, más que nada administrativos, que no son críticos en la ejecución propia de la solución.

La fase de transición es un proceso crítico en el desarrollo del producto, ya que las validaciones finales del sistema son realizadas con los propios investigadores. Para que la transición sea un proceso exitoso, se debe realizar una capacitación exhaustiva al usuario final, basada en manuales de usuarios bien definidos, que son artefactos propuestos por la metodología utilizada, RUP.

Todos los factores propuestos en este punto, validan la utilización de RUP como metodología de desarrollo en esta parte del proceso de desarrollo de la solución.

2.1.3 Desarrollo de la aplicación de procesamiento de salida

La tercera etapa del desarrollo es similar a la primera, el desarrollo de la aplicación cliente, debido a que se deben realizar cálculos finales dependientes de cada proyecto, necesarios para la administración final de los resultados, Siendo la metodología ya definida en la primera etapa, aplicable también a ésta.

2.2 Identificación de Requerimientos

Los requerimientos son obtenidos mediante entrevistas con los investigadores quienes deseen utilizar la capacidad de cómputo brindada. Luego de las entrevistas, y de un análisis de los requerimientos obtenidos, se pasa al primer borrador de la Lista de Requerimientos, los cuales son validados con el investigador y se procede a las primeras definiciones de cómo se llevará el desarrollo de la segunda etapa, mencionada en el punto 2.1.2.

El desarrollo global de la aplicación se enfoca principalmente en el aprovechamiento de la capacidad de procesamiento de cada una de los computadores pertenecientes al grid, así como de la capacidad administrativa con la que el BOINC cuenta, por lo que el desarrollo tiene en cuenta la independencia de trabajo para cada procesador que se incluya al servicio del proyecto. Es decir, el desarrollo es independiente para cada uno de los procesadores propios de una cada una de las computadoras asociadas al grid.

El proyecto podría utilizar software ya desarrollado con anterioridad para cumplir con el procesamiento necesario, por lo que la solución implementada en el grid puede seguir dos caminos para replicar el funcionamiento. La primera es utilizar el software desarrollado y adaptarlo a una solución en grid, mientras que la segunda se trata de desarrollar el procesamiento brindando por el software desde cero, para poder replicarlo.

En el primer camino, el software será administrado por BOINC y se ejecutará en las computadoras asociadas. Para ser lo menos intrusivos en las mismas, debido a que no será un grid dedicado, sino que se tomarán computadores con otros fines principales y se aprovechará su capacidad de cómputo desperdiciado; las aplicaciones no deberían ser instaladas en estos computadores, sino enviadas a cada uno de los clientes, como parte del conjunto de datos de entrada, para su ejecución. Esto sería posible si las instalaciones de cada una de las aplicaciones no son intrusivas en el sistema operativo, y dependen sólo de archivos que se encuentran en sus propias rutas.

En el caso que las aplicaciones utilizadas requieran instalaciones más complejas, se debería tomar la segunda alternativa, e implementar las funcionalidades que estas aplicaciones brindan.

Para que el tiempo de distribución, tanto de los datos de entrada, como de las aplicaciones necesitadas (las cuales tienen mayores probabilidades de representar el mayor porcentaje en cantidad de información enviada) se opta por enviar éstas últimas como un paquete comprimido en formato RAR[G], enviándose también la aplicación necesaria para realizar la descompresión cuando todo el paquete llegue a cada computadora cliente.

2.2.1 Requerimientos funcionales

Los requerimientos funcionales están centrados principalmente en la manipulación de los datos de entrada y salida, así como en el procesamiento en paralelo como tal.

Como se planteó en el punto anterior, la aplicación cliente es la encargada de realizar el procesamiento requerido por cada proyecto de investigación, y dado que se utiliza la técnica de Xtreme Programming, los requerimientos funcionales en este punto son variables a medida que se van cumpliendo objetivos puntuales a corto plazo, dado que se basa en la prueba y error, es posible que cada vez que se alcance un objetivo planteado, nuevos puntos a resolver y nuevos objetivos pueden aparecer a medida que se va desarrollando esta parte de la aplicación. En este punto los requerimientos son definidos en forma iterativa, a medida que el desarrollo vaya cumpliendo los requerimientos planteados en la iteración inicial.

Los requerimientos funcionales definidos como punto base para el comienzo del desarrollo, se toman de la solución planteada por el investigador según el proyecto, es decir, la que se desea optimizar con el desarrollo del grid.

En la segunda etapa del desarrollo, es decir, en el desarrollo de la interfaz Legión, estos requerimientos son definidos y analizados luego de que la primera etapa de desarrollo culmina. Los datos de entrada del grid son obtenidos en esta etapa, además, estos deben seguir algunos patrones y características especiales, para que el procesamiento en la aplicación cliente

sea adecuado. Dado que estos datos son manipulados por el cliente final del grid, éste debe tener conocimiento de los patrones y características que deben tomar los datos de entrada, por lo que se realizan las validaciones necesarias antes de enviar el procesamiento.

2.2.2 Requerimientos no funcionales

Los requerimientos no funcionales son parte activa y fundamental de la solución, ya que la optimización del uso de procesamiento ocioso puede definir que un proyecto de este tipo no sea viable, tanto económica, como funcionalmente.

La aplicación administrativa del BOINC se mantiene bajo un sistema operativo unixlike, particularmente una distribución Linux. La elegida para este proyecto es CentOS 5 dado su extensivo uso en servidores; mientras que la aplicación cliente se mantiene bajo un sistema operativo Windows XP, ya que los laboratorios vienen con este sistema operativo.

La eficiencia es el punto base en el que se apoya la solución, dado que la utilización de recursos, en este caso, capacidad de procesamiento, debe ser manejada de forma eficiente para que la solución propuesta sea válida, y pueda comprobarse que una solución basada en computación en grid, puede ser muy útil para el aprovechamiento de recursos computacionales y que con ellos, se pueda incentivar y proveer de recursos a proyectos de investigación.

La disponibilidad de la aplicación estará limitada por el uso de las computadoras asociadas al grid, ya que, como se dijo anteriormente, la participación de cada computadora en el proyecto no es la función principal que posee.

2.3 Análisis de la Solución

El análisis de la solución contempla tanto el análisis de viabilidad del sistema, teniendo en cuenta aspectos técnicos y económicos; así como la definición de funciones a todos los elementos relevantes del sistema. Además tendremos en cuenta las restricciones que se han identificado, tanto de software y de hardware.

2.3.1 Análisis de Viabilidad

El análisis de viabilidad será disgregado en dos aspectos importantes para determinar si el proyecto debe continuar, si se debe buscar otra alternativa de solución, o en último caso, quedarse con la solución previamente planteada, sin la oportunidad de hacer mejoras en el procesamiento. Estos aspectos son el técnico y el económico.

Aspecto Técnico

El sistema será viable mientras se cuente con un conjunto de computadores que puedan anexarse al grid como computadores cliente, que serán los encargados de procesar las simulaciones. Además de contar con los requerimientos mínimos que la aplicación cliente del BOINC necesite. Estos requerimientos son expuestos en el la tabla 2.1 para el sistema operativo en donde se espera que funcione BOINC.

	Requerido	Recomendado
Sistema Operativo	Windows 98 o superior, Windows 2000 con Service Pack 4	
Procesador	Pentium 233 MHz	Pentium 500 Mhz o superior
Memoria RAM	64 Mb	128 Mb o superior
Disco Duro	20 Mb	

Tabla 2.1 – Análisis de Viabilidad – Aspecto Técnico [5]

Dado que los laboratorios de la propia universidad servirán como computadores cliente y que los procesos deben ejecutarse de forma transparente para el usuario que se encuentra utilizando la computadora, éstos computadores deben tener características muy superiores a las antes mencionada, ya que al tratarse de una universidad, ésta necesita tecnología de punta, la cual superaría largamente los requerimientos mínimos planteados.

Será necesario además, un servidor central quien alojará el servidor BOINC además del servidor de base de datos MySQL.

Aspecto Económico

Los gastos significativos son divididos en dos aspectos: el tecnológico y de equipamiento, y el de recursos humanos.

El aspecto tecnológico incluye el servidor antes descrito, el cual ha sido adquirido por $4000 dólares americanos.

En cuanto a los demás recursos tecnológicos y de equipamiento, se están utilizando tecnologías abiertas para el diseño del grid de computadores, como el desarrollo de la interfaz web del Sistema Legión, necesaria para la automatización de tareas por parte de los usuario. En la tabla 2.2 se muestran todas las tecnologías necesarias.

Aplicación Generadora del Procesamiento	
Lenguaje utilizado	C/C++
IDE utilizado	Microsoft Visual C++ 2005 Express Edition
Procesamiento en Grid	
Plataforma de Software	BOINC – Servidor y Cliente
Base de Datos	MySQL
Interfaz Web	
Lenguaje utilizado	PHP
Visualizador de Internet	Mozilla Firefox 3

Tabla 2.2 – Análisis de Viabilidad – Aspecto Económico

2.3.2 Asignación de Funciones

El desarrollo de la solución conlleva la necesidad de que cada parte que la constituye tengan asignados roles, para que el funcionamiento global sea correcto. La solución está compuesta por cuatro partes, el software, el hardware, los usuarios y la base de datos.

El software

Cuando se culmine el desarrollo de la solución el sistema deberá satisfacer todos los requerimientos que se encontraron en las diferentes reuniones con los investigadores, además de los requerimientos propios de la computación en grid. Además, estos requerimientos deberán ser satisfechos de manera oportuna y eficiente.

Entre las funciones más resaltantes asignadas al software, tenemos la automatización de la ejecución de las tareas, la administración de las mismas, así como el monitoreo y control de las tareas en proceso.

El hardware

El hardware será dividido en dos partes, dependiendo del flujo del proceso a desarrollar. La computadora que tendrá la función de automatizar la creación de tareas y la de servir de interfaz para el usuario que quiera generarlas. Deberá contar con los servicios necesarios, tanto como servidor de aplicaciones, como servidor de base de datos. Además, éste tendrá la función de realizar el procesamiento de los resultados enviados por los computadores asociados al grid. Los computadores que se unirán como clientes al proyecto tendrán la función principal de realizar todo el proceso de ejecución de tareas y luego enviar los resultados al servidor para que ese realice el procesamiento final.

Los usuarios

Los usuarios, específicamente los investigadores, tendrán la responsabilidad de utilizar el sistema para los fines con los que se realizó, por lo que cualquier acción que perjudique el correcto funcionamiento del sistema, o la imposibilidad de que otros usuarios lo utilicen, es responsabilidad del usuario. Por esto, los datos de ingreso al sistema deben ser conocidos de manera exclusiva por el usuario al que se le creó una cuenta dentro del sistema.

La base de datos

La base de datos será el único lugar donde se guarde la información referente al sistema, tanto para la información del BOINC, como para la información necesaria para la administración de usuarios, la generación y administración de tareas.

2.3.3 Definición del Sistema

En la definición de la metodología se define a RUP como parte de la utilizada en la elaboración de la solución, por lo que servirá para la definición del proyecto, mediante toda la documentación generada, tanto en el análisis y el diseño de la misma.

Los documentos necesarios para la definición del problema se encuentran en los anexos. Entre los cuales se encuentran el Documento de Visión, el Documento de Análisis, la Especificación de Requisitos y la Especificación de Requisitos de Software.

Finalizada esta etapa en el proceso de desarrollo del software obtendremos las herramientas que se utilizarán en el resto del proceso descrito, entre las cuales tenemos a los lenguajes de programación, los entornos de desarrollo, los administradores de base de datos, y la tecnología que permite la computacion distribuida.

3 Diseño

En el presente capítulo se presentan los aspectos de diseño desarrollados luego de realizar el proceso de análisis previo. Este punto incluye la arquitectura que presenta la tecnología utilizada, así como la que se plantea para los puntos a desarrollar. Finalmente se presenta el proceso de desarrollo utilizado para el diseño de la interfaz gráfica del sistema.

3.1 Arquitectura de la Solución

La arquitectura elegida para la solución planteada será la referencia base para la construcción del resto del sistema en la etapa posterior al diseño de la

misma. Como en la elección de la arquitectura se basa el resto del desarrollo del sistema, esta arquitectura debe soportar ciertas características que se han visto necesarias para la viabilidad del sistema.[11]

Entre las ventajas en el uso de sistemas distribuidos en los que se apoya la solución planteada en este proyecto de tesis tenemos: [12]

1. Recursos compartidos. Un sistema distribuido permite compartir tanto software como hardware, mediante computadores asociados a través de la red, en este caso, mediante los computadores asociados al grid, cada uno de los cuales brinda capacidad de procesamiento para realizar los procesos requeridos por los investigadores.

2. Apertura. Los sistemas distribuidos están diseñados para soportar equipos de diferentes fabricantes, ya que se basan en protocolos definidos como estándares. Como se ha mencionado anteriormente, la infraestructura BOINC utilizada, se encuentra disponible para diferentes arquitecturas y plataformas, pero para el caso específico del desarrollo, se utilizarán equipos Intel corriendo el sistema operativo Windows XP.

3. Concurrencia. En un sistema distribuido, muchos procesos pueden operar al mismo tiempo en diferentes computadoras de la red. Para el caso específico de la solución planteada, cada computador asociado al grid, se encarga de la ejecución de una parte del proceso.

4. Escalabilidad. Los sistemas distribuidos son escalables, en cuanto nuevos recursos pueden ser añadidos en función a nuevas demandas de capacidad de procesamiento. BOINC sólo necesita que cada computador que desea ingresar al grid cuente con el BOINC cliente y se una al proyecto deseado, indicando la cantidad de recursos que desea compartir.

Así, como se han identificados ventajas en las cuales debemos basar el desarrollo planteado, existen también desventajas inherentes a un desarrollo como el que planteamos realizar.

1. Riesgo de usuarios maliciosos. En el caso de la computación colaborativa en la que se basa el proyecto BOINC, existe el riesgo de la existencia de usuarios maliciosos que alteren los resultados que se procesan en sus computadoras, además de tener el riesgo de que estos usuarios quieran obtener ventajas sobre los beneficios que brinda el proyecto, que para el caso de proyectos con SETI, son créditos. (En función a la cantidad de información que procesa cada usuario). Una forma de minimizar este riesgo es la redundancia de procesamiento, que implica que un proceso sea enviado a más de una máquina para que sea procesada, y dependiendo de si el resultado es similar, éste se toma como válido.

Dado que en nuestro caso, los computadores serán controlados en su totalidad, el riesgo de usuarios maliciosos se minimiza, por lo que la redundancia con la que se podría contar queda descartada ya que no se cree necesaria. Cabe señalar, que si la ejecución de una de las partes en las que se divide el proceso, presentara una falla, BOINC la reconoce y vuelve a enviarla para que se ejecute nuevamente.

3.1.1 La arquitectura BOINC

BOINC presenta una arquitectura cliente-servidor, siendo el servidor BOINC quien ofrece los servicios necesarios para que los clientes BOINC puedan realizar el procesamiento requerido. Estos dos componentes son conectados mediante una red de trabajo que permiten a los clientes acceder a los servicios. La misión de los servidores grid consiste en repartir de manera eficiente las unidades de trabajo, en las cuales ha sido dividida la aplicación, entre el conjunto de nodos que forman el grid. [18]

Los servicios brindados por el servidor BOINC son la administración de los trabajos, procesamiento de resultados parciales y la administración de cuentas de usuario entre los principales; mientras que el cliente BOINC, que se encuentra corriendo en las computadoras participantes, realiza peticiones o ejecuta una aplicación específica al proyecto al que se encuentra unido.

Debido a que BOINC es una tecnología ya implementada con anterioridad, probada y validada por su equipo de desarrollo, la arquitectura que han

utilizado forma parte de la solución propuesta en este documento. Por lo tanto parte de la arquitectura ya no será validada en cuanto a su relación con los requerimientos, ya que se conoce, por proyectos que la utilizan, que los requerimientos relacionados con BOINC son soportados en su totalidad por la misma.

El siguiente paso es realizar una interfaz con el servidor BOINC para que las tareas sean generadas de manera automática, necesitando para esto la elaboración de un sistema Web desde donde se pueda automatizar la ejecución de tareas.

3.1.2 La arquitectura a diseñar

BOINC presenta una arquitectura diseñada y validada previamente, mas se necesitan ciertos componentes adicionales que, en su conjunto conforman la solución planteada.

El siguiente diagrama de bloques muestra la arquitectura global de la solución. Los bloques de color celeste son componentes propios del BOINC, mientras que los bloques de color verde claro son componentes que se pretenden implementar, y que en su conjunto forman la arquitectura de Legión.

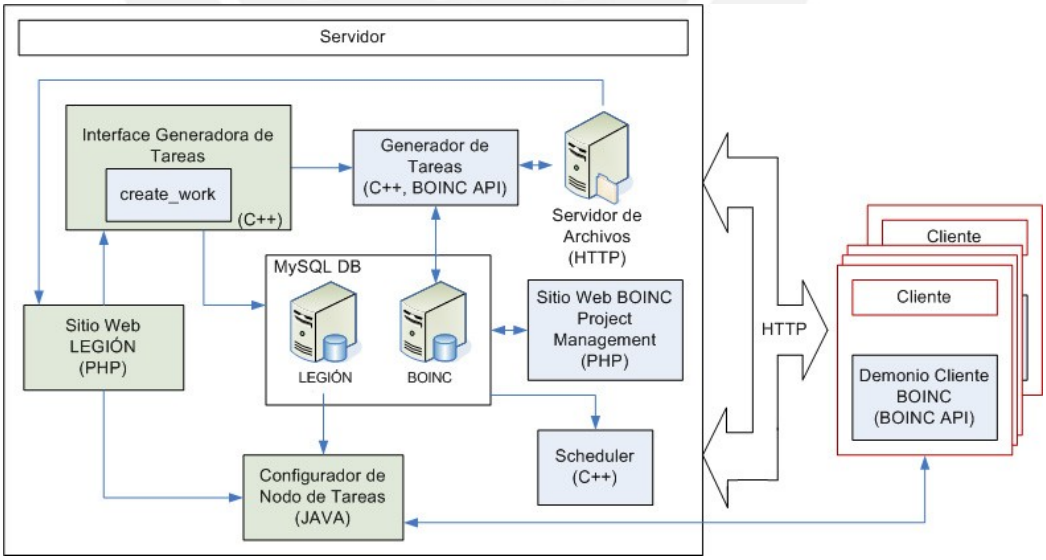

Figura 3.1 – La Arquitectura LEGIÓN

El flujo de trabajo empieza cuando un investigador requiere generar un proceso de cómputo dentro de Legión. Esto se realiza a través del Sitio Web de

Legión, quien invoca a la Interfaz Generadora de Tareas, quien recoge todos los datos obtenidos mediante el Sitio Web, y se encarga de la generación de las tareas invocando a la función “create_work”, propia del API de BOINC. A su vez los datos de la generación de cada tarea son guardados en la base de datos propia del proyecto y en la base de datos propia de BOINC.

El Generador de Tareas tiene, además, comunicación con el Servidor de Archivos, para realizar las comunicaciones con todos los Clientes BOINC instalados en los computadores asociados al proyecto. Esta comunicación se realiza mediante el protocolo HTTP, vía Internet, y además, es supervisada por el “Scheduler”, quien también controla la cola de tareas generadas.

El Configurador de Nodos de Tareas es una aplicación que tiene como finalidad comunicarse con todos los Clientes BOINC y ejecutar procesos administrativos que BOINC nos brinda, como la petición de tareas desde el Demonio Cliente BOINC, o la incorporación o separación de un computador a un proyecto específico. Con esta posibilidad de comunicación, logramos que el tiempo muerto entre las peticiones de tareas sea el mínimo posible, ya que los clientes son quienes realizan las peticiones de tareas de forma automática.

Cuando el procesamiento de una tarea generada finaliza, el Demonio[H] Cliente BOINC se comunica con el Servidor de Archivos, con la finalidad de enviar el archivo resultado del procesamiento. Esto también se realiza mediante el protocolo HTTP.

Luego de realizar el procesamiento de salida, el resultado final del procesamiento estará disponible para su descarga a través del Sitio Web Legión.

La Interfaz Web Legión, quien se encargará de realizar la automatización de la generación de tareas como se ha mencionado antes, será planteada en 4 capas: La capa de Clases, la de Lógica del Negocio, la de Acceso a Base de Datos y la Capa de Presentación.

La Capa de Clases

La capa de clases contiene todo el diseño de clases definidas en la etapa de análisis, y mediante las cuales se pasará la información a las demás capas para realizar los procesos necesarios.

La Capa de Lógica del Negocio

La capa de lógica del negocio es la que se encarga de realizar la mayor carga de procesamiento de información, ya que en ella se encuentran por ejemplo, la lógica de la comunicación con el servidor BOINC para la generación de tareas.

La Capa de Acceso a Base de Datos

La capa de acceso a base de datos se encarga de realizar las conexiones con la base de datos, para luego realizar las peticiones de información, o los registros de la misma, según se requiera. A esta instancia se llega luego de pasar siempre por la capa de lógica del negocio.

La Capa de Presentación

La capa de presentación se encargará de mostrarle al usuario el flujo del registro de una nueva petición de procesamiento, además podrá visualizar todas las tareas que ha generado con anterioridad, y el estado de ejecución de las mismas. El procesamiento de la información que el usuario ingresa a este nivel, se remite a validaciones simples, tratando de minimizar los errores por parte de los usuarios. No se pretende realizar procesamientos complejos a este nivel.

3.2 Diseño de la Interfaz Gráfica

Se han tomado ciertos principios de diseño de interfaz gráfica que brindan la base con la que todo sistema debe contar para realizar un buen diseño de la misma. Por ejemplo, los términos y conceptos utilizados en la Interfaz Gráfica son familiares a las personas que utilizarán el sistema, además, los objetos manipulados por ésta deben estar directamente relacionados con el ambiente de trabajo del usuario. Para nuestro sistema se utilizan algunos términos

relacionados con el proyecto de investigación que se requiera, íntimamente relacionados con los usuarios finales esperados del sistema.

Los usuarios esperados en el sistema, son usuarios avanzados, por lo que se asume que ya han tenido experiencia con diferentes tipos de interfaz gráfica. Debido a esto la consistencia en la interfaz gráfica será un punto fundamental en que el tiempo de aprendizaje por parte de los usuarios sea el mínimo posible.

Esta interfaz debe proveer al usuario de mecanismos que les permitan recuperarse frente a posibles errores. Es utópico pensar que un usuario esté libre de cometerlos, debido a esto, el sistema debe poseer mecanismos para minimizar su ocurrencia. Debemos tener en cuenta que los errores no pueden ser totalmente eliminados, por lo que la interfaz debe brindar las facilidades para que los usuarios puedan recuperarse frente a ellos. El diseño propuesto presenta confirmaciones de acciones antes de ejecutarlas, mostrando siempre los valores que el usuario ha ingresado para que pueda verificarlos. [12]

Finalizada esta etapa en el desarrollo del proyecto se obtiene la arquitectura final a utilizar en el desarrollo del proyecto, la cual estará en la Especificación de Requisitos de Software.

Las imágenes 3.2 y 3.3 muestran el prototipo de la interfaz Legión, especificamente la pantalla inicial del sistema con el login de usuario, y luego la pantalla de administración de tareas.

Figura 3.3 – Administrador de Tareas de Legión

4 Construcción

En el presente capítulo se desarrollan los aspectos propios de la construcción de la solución planteada en este documento. Se describe el proceso de desarrollo para cada una de las partes en las que se dividió el proyecto, justificando las herramientas utilizadas en cada uno de ellos.

4.1 Construcción

Los procesos principales que posee el sistema, están relacionados con la generación de tareas, para que los clientes BOINC sean los encargados de procesarlas y enviar los resultados hacia el servidor BOINC. El desarrollo de este proceso consta de tres partes bien diferenciadas, tanto en la forma de desarrollo, como en las tecnologías y estilos de programación que cada parte involucra.

4.1.1 Procesamiento Inicial

El primer punto del desarrollo consta de la aplicación que será enviada a los clientes y que contiene el procesamiento requerido por el investigador. Este procesamiento inicial puede tener dos tipos de implementaciones. Como se ha visto anteriormente, el BOINC brinda la posibilidad de utilizar el API propio de BOINC, y además da la posibilidad de utilizar una aplicación propia de BOINC, llamada “wrapper”, que es la que ejecuta una aplicación como un subproceso.

Debido a que el BOINC nos brinda la posibilidad de compartir con el proyecto, uno o más de los procesadores con los que cuenta cada computador asociado al mismo, y, dado que cada proceso es similar, por lo que en muchas

ocasiones se deben de compartir recursos, generalmente archivos de entrada, se debe separar cada proceso, de tal forma que los recursos puedan ser compartidos en forma paralela. La réplica de recursos, para que cada proceso utilice los suyos propios, es la única opción viable. Con esto validamos el caso en que la aplicación necesite permisos de escritura sobre los recursos mencionados.

Como se ha mencionado, el proyecto realizará el proceso sobre computadores que tienen otro papel fundamental, y se utilizará su capacidad de procesamiento ociosa cuando se necesite. Debido a que el procesamiento planteado no es la tarea fundamental de cada computador, el usuario potencial sobre éste no debe sentirse afectado por el procesamiento, por lo que es necesario que el proceso se ejecute en modo “background”, y además, los recursos que utilice deben ser controlados de manera eficiente, tanto para que el usuario pueda seguir con sus labores, como para que el proceso se ejecute de forma oportuna.

Para realizar todo este procesamiento previo se eligió hacer el desarrollo sobre el lenguaje C, debido a que se necesitaba un procesamiento no muy complicado y a bajo nivel, donde la velocidad de procesamiento es fundamental, dados los objetivos del proyecto.

4.1.2 Interfaz Web

La implementación del generador de tareas pueden realizarse siguiendo dos métodos posibles. La primera de ellas es la invocación de una aplicación previamente desarrollada, la cual recibe cierta cantidad de parámetros con los que cada unidad de trabajo es generada. Cabe señalar que esta aplicación que proporciona BOINC esta implementada bajo el API de BOINC, por lo que se encuentra desarrollada en el lenguaje C++. La segunda opción es la utilización del API de BOINC y generar una aplicación similar a la descrita anteriormente, con la salvedad de que sería diseñada e implementada en base a los requerimientos específicos propios del proyecto.

Para el sistema en desarrollo, se eligió realizar el generador siguiendo el segundo método descrito, implementando el generador mediante el API de BOINC, logrando una aplicación personalizada y eficiente.

La interfaz web será la encargada de realizar todo el procesamiento necesario para que la generación de tareas se realice de manera exitosa. Para el desarrollo de esta interfaz se tenían tres herramientas de programación web: PHP, Asp.NET y JSP como candidatas.

En el análisis comparativo realizado en la Universidad del Norte en Barranquilla, Colombia, se abarca la portabilidad de estas herramientas, la confiabilidad, la arquitectura de hardware y software, detección de fallas y la complejidad de la programación. [14]

Estos análisis se llevaron a cabo en los sistemas operativos Windows 2000 Server y Linux Red Hat 7, ambos sistemas operativos diseñados para fungir como servidores, y, con el software necesario, como Servidores Web.

Portabilidad en los sistemas operativos

Como el estudio antes mencionado indica, las tres herramientas mencionadas son portables tanto de sistemas Windows a sistemas Linux, como viceversa. La eficiencia con la que estas herramientas trabajan, en cada sistema operativo, si se ve mermada por la migración. Asp.NET es un producto diseñado para trabajar sobre plataformas Windows, siendo necesaria la herramienta One Active Server Page desarrollada por SUN para poder correr bajo Linux. Esta herramienta es propietaria, por lo que la utilización de Asp.NET sobre Linux se reduce.

La tabla 4.1, tomada del estudio mencionado, muestra las tres herramientas de desarrollo y los sistemas operativos donde los códigos pueden ejecutarse, mencionando si es que se necesita un software adicional.

Herramienta	Apache		IIS		Tomcat		OneASP	
	Win 2000	Linux	Win 2000	Linux	Win 2000	Linux	Win 2000	Linux
PHP	X	X	X					
Asp.NET			X					X
JSP	X	X			X	X		

Tabla 4.1 – Pruebas de Portabilidad [14]

Confiabilidad

Las tres herramientas se ejecutan del lado del servidor, por lo que al seleccionar la opción Ver Código Fuente, disponible en el navegador de Internet, ninguna de las herramientas muestra el código del lado del cliente.

Arquitectura de Software y Hardware

Dado que, tanto PHP como JSP, fueron diseñados para trabajar sobre plataformas Linux, el rendimiento de éstas sobre Windows es inadecuado. El mismo criterio se aplica sobre Asp.NET, que fue diseñado para trabajar bajo Windows, por lo que en este sistema operativo tendrá un mejor rendimiento. Las características asociadas a cada herramienta para trabajar correctamente se muestran en la tabla 4.2.

	Herramientas		
Características necesarias para un funcionamiento adecuado	PHP	Asp.NET	JSP
Sistema Operativo	Linux	Windows	Linux
Servidor	Apache	IIS	Tomcat
Memoria	128 Mb o más	128 Mb o más	256 Mb o más

Tabla 4.2 – Arquitectura de software y hardware [14]

Detección de Fallas

En el estudio citado, se realizaron procesos en paralelo para cada una de las herramientas, y se observó la forma como la herramienta detectaba los tipos de errores producidos. Los resultados citados han sido rediseñados para una mejor visualización en la tabla 4.4.

	Herramientas		
Detección de Fallas	PHP	Asp.NET	JSP
Óptimo		X	
Medio	X		
No óptimo			X

Tabla 4.3 – Grado de detección de fallas [14]

La tabla 4.4 muestra la cantidad de errores detectados por cada una de las herramientas, de un total de 150 errores generados, en promedio.

Herramienta	Hay error + Ubicación	Tipo de error
PHP	135 (90%)	23 (15%)
Asp.NET	83 (55%)	69 (46%)
JSP	143 (95%)	140 (93%)

Tabla 4.4 – Calidad de fallas detectadas [14]

Luego de producidos los errores, se verificó la Base de Datos, y se analizó la consistencia de Base de Datos que brindaba cada una de las herramientas, de donde se obtienen los resultados mostrados en la tabla 4.5.

Herramientas	Integridad con la Base de Datos MySQL	
	Windows	Linux
PHP	132 (88%)	141 (94%)
Asp.NET	71 (47%)	68 (45%)
JSP	67 (46%)	74 (49%)

Tabla 4.5 – Integridad de Base de Datos [14]

Complejidad en la Programación

En el estudio referenciado, se realizaron pruebas sobre un prototipo de software con los módulos de prueba cliente, vendedor, y artículos. Los cuales fueron elegidos por el grupo investigador como un sistema transaccional estánndar. La complejidad de la programación se medirá en base a las líneas de código a ejecutarse en la realización de transacciones estándar.

	PHP	Asp.NET	JSP
Actualizaciones			
Artículo	41	43	66
Cliente	39	39	70
Vendedor	39	39	73
Inserciones			
Artículo	15	18	31
Cliente	12	18	31
Vendedor	12	19	37
Consultas			
Artículo	31	48	71
Cliente	31	40	66

Vendedor	31	40	68
Listado	11	20	28
Eliminaciones			
Artículo	29	69	74
Cliente	28	53	71
Vendedor	28	60	70
VENTA	231	291	299

Tabla 4.6 – Complejidad de Programación [14]

Dados los resultados mostrados en la tabla 4.6, el equipo investigador no encontró diferencia estadística significativa, y atribuyó las diferencias en la cantidad de líneas a las instrucciones obligatorias en unos lenguajes de programación, que son de carácter opcional en los otros.

Los promedios en los tiempos de respuesta obtenidos en cada una de las transacciones realizadas se muestran en la tabla 4.7.

	PHP		**Asp.NET**		**JSP**	
Actualizaciones						
	Linux	Win2000	Linux	Win2000	Linux	Win2000
Artículo	0.0079	0.3949	0.3124	0.2121	0.0029	0.1772
Cliente/Vendedor	0.0081	0.4270	0.3902	0.4996	0.0036	0.1807
Inserciones						
Artículo	0.0070	0.2262	0.2456	0.1057	0.0028	0.0677
Cliente/Vendedor	0.0109	0.2212	0.1074	0.1030	0.0052	0.0458
Consultas						
Artículo	0.0056	0.1430	0.1244	0.1007	0.0024	0.0927
Cliente/Servidor	0.0061	0.1975	0.2596	0.3096	0.0034	0.0949
Eliminaciones						
Artículo	0.0314	0.6694	0.1123	0.1612	0.0104	0.2162
Cliente/Vendedor	0.0314	0.3378	0.5352	0.5184	0.0122	0.1744
Venta	0.0398	0.3945	0.2860	0.3005	0.0181	0.1100
Consulta de Listado	0.2228	6.3581	0.9455	7.7026	0.0324	0.1126

Tabla 4.7 – Promedios de tiempo de respuesta [14]

Como era de esperarse, las herramientas diseñadas para un sistema operativo específico, poseen tiempos de respuesta inferiores en dicho sistema operativo.

Como el estudio lo indica, no se puede afirmar que una herramienta sea mejor que otra, por lo que la decisión de utilización se debe basar en las características que se necesitan sean soportadas por un desarrollo específico.

Dadas las características del desarrollo que se plantea como solución en este documento, tanto la arquitectura de software y hardware como la integridad que presenta el lenguaje con la base de datos son aspectos críticos del desarrollo.

La arquitectura de los servidores en los cuales se pretende poner en producción al proyecto, se basa en el sistema operativo CentOS 5, una distribución Linux, con el Servidor Web Apache. La integridad con la base de datos antes la ocurrencia de cualquier error es un factor fundamental, ya que la generación de tareas, el control del estado de la ejecución de las mismas, y el procesamiento de salida, dependen de procesos involucrados directamente con la base de datos.

4.1.3 Procesamiento de Salida

El procesamiento de salida recoge todos los resultados parciales obtenidos en cada procesamiento enviado a cada máquina asociada al grid y realiza el procesamiento final, que consta de la lectura de los resultados obtenidos en archivos de texto, para finalmente realizar resúmenes según el investigador los requiera. Además, para un manejo más eficiente de la capacidad de almacenamiento del servidor, los archivos con los resultados parciales serán eliminados luego del procesamiento final.

La aplicación que realiza el proceso descrito se ejecuta mediante una cascada de disparadores (triggers) desde la base de datos. El disparador final será el encargado de verificar cuando una tarea ha sido finalizada, es decir, cuando cada una de las unidades de trabajo, pertenecientes a la tarea, finaliza.

Finalizado este proceso, el resumen generado estará disponible para la descarga del usuario quien lo ha generado.

El procesamiento final es muy similar al procesamiento inicial, por lo que el desarrollo también será realizado en lenguaje C.

4.2 Pruebas

Las pruebas son necesarias en todo proceso de desarrollo de software, tanto para validar que los requerimientos definidos en la etapa de análisis sean cumplidos por el software, como para validar la existencia de errores, para que luego sean corregidos.

En el primer punto del desarrollo y en el último, las pruebas son realizadas a medida que se avanza en el desarrollo de las dos aplicaciones mencionadas, dado que se necesita validar el correcto funcionamiento de lo ya desarrollado para seguir con el proceso.

Las pruebas unitarias en este punto serán fundamentales, dado que estas enfocan sus esfuerzos de verificación en unidades pequeñas del software [19], validando, como ya se mencionó, que la etapa en proceso de desarrollo sea validad, ya que sin ésta, el proceso de desarrollo no puede continuar.

Luego del desarrollo de todos los módulos, y las respectivas pruebas unitarias, se realizarán pruebas de integración thread-based testing [19], las cuales se basan en integrar el conjunto de clases requeridas para responder a cierto evento producido en el sistema, siendo cada hilo integrado y probado individualmente.

Las pruebas de la Intefaz Legión serán realizadas en diferentes escenarios, lo que permitirá realizar validaciones sobre el comportamiento de la misma, así como validar los resultados obtenidos luego de finalizado el proceso de generación y ejecución de tareas.

Las pruebas principales, las cuales validarán la eficiencia y desempeño del sistema, son las pruebas de rendimiento. Estas pruebas son diseñadas para probar el tiempo de ejecución del sistema, dentro de un escenario válido con el sistema integrado. [18]

Se necesita saber, además, la capacidad límite de procesamiento del sistema, frente a la generación masiva de tareas, por lo que pruebas de estrés serán necesarias. Las pruebas de estrés están diseñadas para confrontar al

sistema a situaciones anormales [19], en este caso, en la generación masiva de tareas, incrementando el número de tareas generadas, hasta uniformizar los resultados obtenidos, y luego, realizar incrementos en el número de tareas de manera gradual, siendo necesario además, que el número de nodos asociados al grid, varíe en cantidad según el número de tareas generadas. Dado esto, para las pruebas de estrés se manejan dos variables: la cantidad de tareas a generar y el número de nodos asociados al grid.

Al finalizar esta etapa en el desarrollo del proyecto se obtendrán primero las aplicaciones de procesamiento de entrada como de salida, los prototipos de la interfaz web necesaria para la generación de tareas y el diseño de las pruebas que se pretenden realizar sobre el sistema.

5 Caso de Aplicación

En el presente capítulo se presenta el caso de aplicación que muestra la forma como se implemento un proyecto de investigación dentro del Sistema Legión. Primero se plantea el problema específico de este proyecto de investigación, para luego mostrar la forma como se resuelve el problema.

5.1 Proceso de Simulación Bayesiana

En la literatura estadística, varios modelos de regresión binaria, abordados desde el punto de vista de la inferencia bayesiana [1], han sido propuestos, muchos de los cuales se encuentran aún en estudio, para una posterior validación y aceptación, como el propuesto por Bazán et al, [33] denominado BBB Skew Probit.

Este tipo de inferencia estadística se basa en simulaciones Monte Carlo con Cadenas de Markov, realizando cálculos vía simulación muy complejos, donde además, el diseño de la simulación es generalmente para un número elevado de casos, por lo que el recurso computacional es fundamental para su resolución. El tiempo requerido para la resolución de este tipo de simulaciones es muy elevado, aun con ayuda computacional, por lo que se plantea una solución que mejore los tiempos de ejecución sustancialmente. Como veremos en la tabla 5.1, el tiempo de ejecución de 1000 data sets puede tomar mas de 11 horas de procesamiento.

5.2 Definición del Problema

El caso de aplicación planteado se enmarca dentro de la resolución de problemas estadísticos, específicamente modelos dicotómicos vinculados al análisis bayesiano. El modelo que se ha tomado como base para el caso de estudio es llamado BBB Skew Probit (Bazán-Bolfarine-Branco), teniendo en consideración una serie de modelos más, los cuales formarán parte de este estudio.

A pesar de que algunos de estos modelos están implementados en programas estadísticos comerciales como el SPSS, hay otros modelos propuestos que por el momento son teóricos, incluyendo su solución vía inferencia bayesiana, por lo que requieren ser probados en diferentes escenarios para poder llegar a una conclusión certera sobre la utilidad de los mismos. Esto obliga el uso de programas estadísticos intermedios como R[3] y OpenBUGS[2], los cuales han sido recopilados en BRMUW [Proyecto DAI 3412].

Para la resolución de estos modelos se necesitan realizar muchas simulaciones, y en diferentes etapas bien definidas, las cuales están basadas en la realización de cálculos matriciales muy complejos. Estos cálculos están basados, en gran parte, en procesos de simulación, tanto para la generación de datos de entrada, como para el proceso principal de verificación en sí.

Como punto de partida, se deben realizar simulaciones de matrices muy grandes, cada una de ellas conteniendo los datos iniciales con los cuales se probará el modelo estudiado. Todas estas matrices, utilizadas para realizar las simulaciones, poseen grados muy elevados, además, son necesarias numerosas iteraciones para poder probar que el modelo resuelve un tipo de problema en particular, que sumado con la cantidad de datos generados en las matrices, dan una mayor confianza de que el modelo es válido para la resolución de esta clase de problemas.

Todos estos factores son los que lo dificultan en gran medida el proceso de cálculo, por lo que es necesario el apoyo de tecnología computacional para su resolución.

Un objetivo importante al momento de la implementación de los modelos estadísticos es, que si se llega a probar que cada uno de ellos resuelve cierta

clase específica de problemas, estos sean utilizados por personas interesadas en el tema, y en la medida que empiecen a ser difundidos, sean finalmente aceptados.

Para la validación de los modelos implementados en BRMUW[I] se necesitan seguir ciertos pasos, los cuales se detallan a continuación. Se tomará como base el modelo Skew-Probit.

- Fijar los valores iniciales de β, n (sujetos) y k (variables aplicadas).
- Simular datos que sigan el modelo Logit.
- Correr la sintaxis del modelo Logit en OpenBUGS [2] utilizando los datos simulados en el paso anterior. Esto se realiza mediante simulación MCMC (Markov Chain Monte Carlo) [34], y se aplicará un parámetro B que indica la cantidad de simulaciones a realizarse.
- Resumir las estadísticas de los β's obtenidos mediante la simulación realizada en el paso anterior.
- Comparar los β's fijados en el paso inicial con los β's obtenidos en el paso anterior. Lo favorable sería obtener una distancia muy cercana a cero entre los β's obtenidos y comparados.

Donde la cantidad de sujetos, y las variables aplicadas determinan el tamaño de las matrices a utilizar en las simulaciones posteriores. A este conjunto de matrices las llamaremos data sets. Se utiliza el programa R [3] para realizar el proceso de simulación de las matrices de datos iniciales, dado que contiene todas las funciones estadísticas necesarias ya implementadas.

Este proceso de generación de data sets se realiza una cantidad predeterminada de veces para el conjunto de datos simulados en el paso 2, al cual llamaremos N.

El tiempo de ejecución depende de ciertos factores propios del modelo como de las simulaciones realizadas. Estos factores son presentados a continuación:

1. n: Número de sujetos.

2. k: Número de variables, que junto con el número de sujetos, determinan el tamaño de las matrices utilizadas, las cuales conocemos como data set's.
3. B: Número de simulaciones MCMC[34] aplicadas a los data set's.
4. N: Número de data set's.
5. Complejidad propia del modelo.
6. Sintaxis del programa a ejecutar.

Cada una de estas simulaciones estan asociadas a un juego de archivos, uno conteniendo el estudio de simulación, el archivo con extensión .R, y otro conteniendo el modelo a estudiar, el archivo con extensión .BUG (extensiones utilizadas por el software estadístico R y el software estadístico OpenBUGS).

5.3 Descripción de la Solución

El primer punto del desarrollo consta de la aplicación que será enviada junto con el wrapper.C, que se encarga de realizar el procesamiento de los archivos .R y .BUG, con los que se dan inicio las simulaciones. Este procesamiento inicial fija las rutas de instalación de las aplicaciones R y OpenBUGS [2] dentro de cada computadora asociada al grid. Esta instalación es temporal, ya que las aplicaciones mencionadas son enviadas al cliente en modo comprimido, teniendo que realizar el proceso de descompresión, una vez en el cliente, en las carpetas propias del BOINC, por lo que el mapeo de rutas antes mencionado refiere a carpetas administradas por BOINC.

Luego del procesamiento inicial sobre los archivos, se deben redirigir hacia el programa R, quien es el encargado de realizar las simulaciones con ayuda del OpenBUGS.

Dado que el proceso MCMC[34] es fuertemente relacionado, ya que cada uno de los valores que se generan en la cadena, dependen del valor generado anteriormente, y para poder realizar el proceso de simulación de forma distribuida, el parámetro N, que representa la cantidad de data set's generados, será la forma como paralelizaremos el proceso.

Cada “data set” representará una unidad de trabajo, quienes en conjunto, representarán la tarea a generar mediante el Sistema Legión. El siguiente gráfico muestra al sistema pidiendo los valores necesarios para generar una simulación.

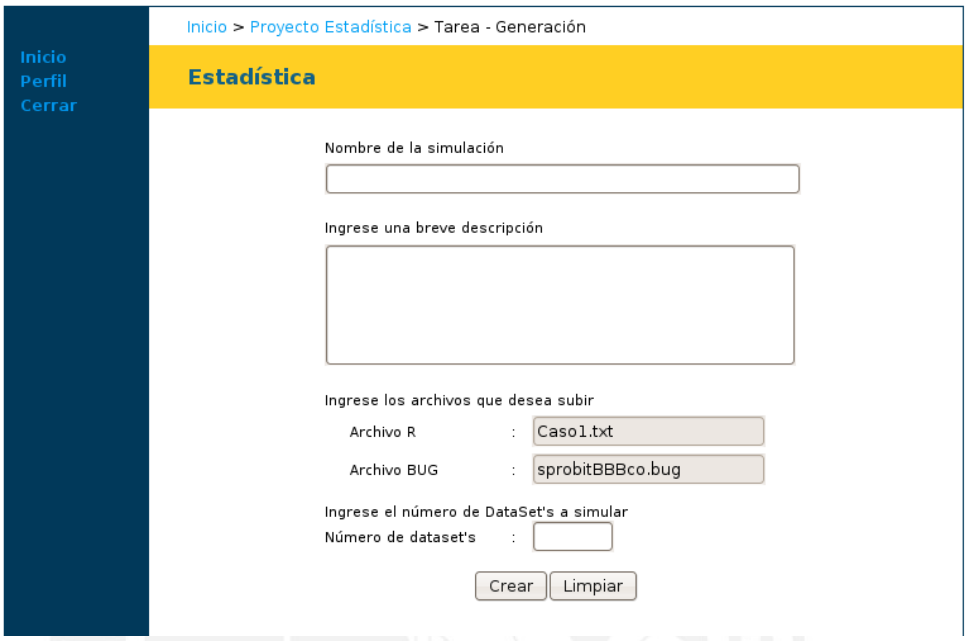

Figura 5.1 – Pantalla de Generación de Tareas - Legión

Como muestra la imagen, el sistema pide la cantidad de data set's que el usuario desea generar en este particular proceso de simulación. Para proporcionar al sistema una medida adicional de seguridad, la primera vez que el usuario desee generar una simulación asociada a un par de archivos específicos, .R y .BUG, esta solo estará conformada por un data set, pudiendo generar mayores cantidades luego de que la tarea inicial termine de ejecutarse de manera exitosa.

El procesamiento de salida realiza los resúmenes correspondientes, según las necesidades del investigador, y el proceso seguido es el que se describe en el capítulo previo.

Inicialmente, el proceso de simulación estadístico se realizaba de forma secuencial utilizando un solo computador de escritorio, y los tiempos de ejecución tomados para una cierta cantidad de data sets es la siguiente:

Antes (Sin Legión)

DataSets	Tiempo
1	00:00:42
4	00:02:46
5	00:03:28
10	00:06:56
100	01:09:28
1000	11:34:41

Tabla 5.1 – Solución Previa – Sin Legión

Luego de la implementación de la solución en este escenario específico, los tiempos de procesamiento se ven reducidos de manera significativa, los resultados pueden ser vistos en la siguiente tabla:

Despues (Con Legión)	
DataSets	Tiempo
100	00:06:11
250	00:11:37
500	00:20:26
1000	00:36:01
2500	01:24:59
5000	02:56:07

Tabla 5.2 – Solución final – Con Legión

El siguiente gráfico muestra la relación de los tiempos de procesamiento para ambas soluciones, teniendo en cuenta el incremento de los data set's a ejecutar.

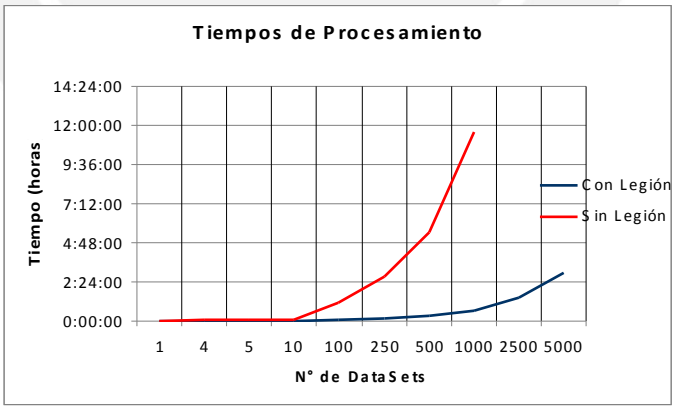

Figura 5.2 – Relación de Tiempos de Procesamiento

Cabe señalar que para la realización de esta comparativa se utilizaron un mismo tipo de computadores, para evitar que los tiempos de ejecución varien por efectos de procesadores más veloces. El modelo del computador utilizado fue un Intel Core 2 Duo 6300 1.86GHz, teniendo un total de 31 computadores.

6 Observaciones, conclusiones y recomendaciones

En el presente capítulo se presentan las observaciones, conclusiones y recomendaciones que derivan del desarrollo de este documento de tesis. Se recalcarán los aspectos del proyecto más relevantes a los cuales se les desea dar énfasis, los resultados consecuencia del trabajo realizado, las conclusiones a las que se llegó y las posibles ampliaciones factibles que se le pueda dar al estudio.

6.1 Observaciones

Se ha pretendido sentar las bases para apoyar a proyectos de investigación, tanto en metodologías de desarrollo de proyectos, como con el proyecto mismo, ya que se espera que procesos computacionalmente intensivos, y que puedan ser procesados de forma paralela, sean anexados dentro del Sistema Legión, pudiendo conseguir mejoras de rendimiento significativas.

BOINC no resuelve todos los problemas de cómputo intensivo, sino sólo aquellos que puedan ser divididos en tareas más pequeñas e independientes unas de otras. El primer análisis a realizar al empezar la construcción de un nuevo módulo para el Sistema Legión, se debe entender claramente la problemática y la forma como se soluciona ésta, específicamente, en como se implementa el algoritmo de solución, para luego realizar el análisis y determinar si este algoritmo puede ser ejecutado de forma paralela.

El uso de software libre multiplataforma es una excelente manera de ahorrar recursos financieros en la universidad, ya que podemos acceder a herramientas bastante productivas y útiles para investigadores de diversas áreas, sin incurrir en costos por el pago de licencias. Además, cabe señalar, que muchas de las herramientas comerciales más importantes en diferentes áreas, tienen una contraparte de distribución gratuita, igualmente funcional y de buena calidad.

6.2 Conclusiones

Con el trabajo realizado en el presente proyecto, se han investigado y analizado diversas herramientas con las cuales se puede realizar computación paralela, tanto infraestructuras ya definidas e implementadas, como diversas librerías las cuales soportan el desarrollo en paralelo. Cada una de estas tecnologías pueden ser aplicadas en distintos escenarios, y resolver un problema de diversas maneras, pero se conoce también, que para un escenario específico, cada una de estas tecnologías posee ventajas y/o desventajas sobre las demás, por lo que se debe realizar un análisis exhaustivo de cada una de estas herramientas enmarcadas en el escenario a resolver, para luego elegir la más factible.

La importancia en la capacidad y velocidad de procesamiento en la resolución de procesos diversos es fundamental en un desarrollo automatizado. Muchos de estos procesos necesitan tiempos cortos de procesamiento, otros procesos pueden soportar tiempos medianamente largos, pero cuando se obtienen tiempos muy largos de procesamiento, estos pueden ser disminuidos utilizando procesamiento en paralelo, cuya difusión estos últimos tiempos ha crecido, y se está utilizando en diversos proyectos, tanto de investigación, como en procesos empresariales.

La infraestructura BOINC puede ser adaptada a distintos procesos, debido a que posee su propia API de desarrollo, la cual puede ser incluida en aplicaciones ya implementadas, con ciertos cambios de lógica, pero no muy significativos para un desarrollador experimentado. Procesos no dependientes de BOINC pueden ser “boincificados” y ejecutarse en paralelo sin necesidad de utilizar el API de BOINC de manera directa.

Entre los procesos candidatos para su ejecución en entornos paralelos podemos contar con aquellos que presenten gran cantidad de ciclos iterativos, o aquellos que trabajen de forma extensiva con matrices, realizando operaciones como sumas o multiplicaciones.

El desarrollo de un sistema Web permite la difusión rápida y masiva de los proyectos de investigación involucrados en el presente proyecto de tesis, lo cual permite que muchas personas, tanto investigadores, docentes y alumnos, se beneficien con los resultados del mismo.

6.3 Recomendaciones y trabajos futuros

Como recomendaciones finales, se esperaría que procesos que necesiten una alta capacidad de procesamiento, sean analizados para estudiar la factibilidad de ser “boincificados” y que puedan contar con capacidad de procesamiento ociosa, con lo que se lograría una mayor capacidad y velocidad de procesamiento.

Un trabajo futuro factible, es el desarrollo de un proyecto de administración de todas las computadoras asociadas al grid, con lo que se tendría un acceso remoto sobre cada cliente BOINC instalado en todas estas computadoras. Este proceso administrativo permitiría un mayor control sobre las computadoras asociadas al grid, la logística y tiempo necesarios para esta administración, se vería reducido.

Además, se pretende trabajar con un servidor de datos externo, debido a que algunos proyectos podrían generar archivos resultado bastante extensos, por lo que un mismo servidor sería insuficiente e ineficiente.

El desarrollo del Sistema Legión tiene como objetivo captar a la mayor cantidad de investigadores para que hagan uso de éste, y puedan terminar sus proyectos de investigación de forma satisfactoria. Dado que los computadores de los laboratorios utilizan el sistema operativo Microsoft® Windows® XP, se esta trabajando en lograr que éstos puedan a la vez soportar un sistema operativo GNU/Linux, trabajándolos como máquinas virtuales corriendo sobre

Microsoft® Windows®. Con esto logramos tener el soporte para aplicaciones nativas tanto en Microsoft® Windows® como en GNU/Linux.

Glosario de Términos

[A] XML : Extensible Markup Language.

[B] SOAP : Simple Object Access Protocol

[C] WSDL : Web Services Description Language.

[D] PVM : Parallel Virtual Machine.

[E] MPI : Message Passing Interface.

[F] POSIX : Portable Operating System Interface.

[G] RAR : Formato de archivo privativo con algoritmo de compresión sin pérdida.

[H] Demonio : Programa que está ejecutándose continuamente en un computador.

[I] BRMUW : Proyecto DAI 3412.

Bibliografía

[1] Jonson, V. E. & Albert J. H (1999). Ordinal Data Modeling. Springer, New York.

[2] The BUGS Project http://www.mrc-bsu.cam.ac.uk/bugs/

[3] The R Project http://www.r-project.org/

[4] Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar, “Introduction to parallel Computing”, 2da edición. Pearson, Harlow 2003.

[5] BOINC http://boinc.berkeley.edu

[6] SETI@Home http://setiathome.berkeley.edu/

[7] EINSTEIN@Home http://einstein.phys.uwm.edu/

[8] Rosetta@Home http://boinc.bakerlab.org/rosetta/

[9] Extreme Programming http://www.extremeprogramming.org/

[10] Rational Unified Process http://www-306.ibm.com/software/rational/

[11] Kroll, Per, “The rational unified process made easy: a practitioner's guide to the RUP”, 1ra edición. Addison-Wesley, Boston, 2003.

[12] Sommerville, Ian, “Software engineering”, 7ma edición. Pearson, Addison Wesley, Boston 2004.

[13] 2006. Comentario del 17 de febrero del a “BOINC.BOINC API (Short Introduction)”. Consultada 02 de julio del 2009. <http://are.ehibou.com/boinc-boinc-api/>

[14] Daladier Jabba Molinares, Adalgisa Alcocer Olaciregui, Carmenza Rojas Morales. 2004. “Análisis Comparativo de las herramientas de programación Web: PHP, Asp y JSP, bajo los sistemas operativos Linux y Windows”, Universidad del Norte. Consultada 02 de julio del 2009. <http://dialnet.unirioja.es/servlet/articulo?codigo=2508544&orden=145248&info=link>

[15] E.H. D'Hollander, G.R. Joubert, "Parallel Computing Fundamentals and Applications", Procceedings of the International Conference ParCo99. Imperial College Press, England 2000.

[16] Smith, Roger. 2004. "Grid Computing: A Brief Technology Analysis". Consultada 02 de julio del 2009. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.109.943&rep=rep1&type=pdf>

[17] Ian Foster, Carl Kesselman, "The Grid: Blueprint for a New Computing Infraestructure", 1ra. edición. Morgan Kaufmann Publishers, California 1998.

[18] Pedro Morillo Tena, “Grid Computing: Compartición de recursos y optimización del hardware”, Mundo Electrónico N° 346, pp. 50-55, Oct. 2003.

[19] Roger Pressman, “Software Engineering: A Practitioner’s Approach”, 6ta edición. Mc Graw Hill, New York 2005.

[20] Ahmar Abbas, “Grid Computing : A Practical Guide to Technology and Applications”, 1ra edición, Charles River Media, Massachusetts 2004.

[21] Sergei Gorlatch, Paraskevi Fragopoulou, Thierry Priol, “Grid Computing : Achievements and Prospects”, 1ra edición, Springer, 2008.

[22] Ian Foster, Carl Kesselman, “The Grid 2: Blueprint for a New Computing Infrastructure”, 2da edición, Morgan Kaufmann, 2003.

[23] El-Ghazali, Albert Zomaya, “Grid Computing for Bioinformatics and Computational Biology”, 1ra edición, Wiley-Interscience, 2007.

[24] Lee-hing Yan, Lee Bu Sung, Teo Yong Meng, “GCA 2007: Proceedings of the 3rd International Workshop on Grid Computing and Applications, Biopolis, Singapore, 5-8 June 23007”, 1ra edición, World Scientific Publishing Company, 2007.

[25] Peter Kacsuk, Robert Lovas, Zsolt Nemeth, “Distributed and Parallel Systems: In Focus: Desktop Grid Computing”, 1ra edición, Springer, 2008.

[26] http://www.terracottatech.com/

[27] Terracota Inc., “The Definitive Guide to Terracotta: Cluster the JVM for Spring, Hibernate and POJO Scalability”, 1ra edición, Apress, 2008.

[28] Pawel Plaszczak Jr., Richard Wellner, “Grid Computing: The Savvy Manager's Guide”, 1ra edición, Morgan Kaufmann, 2005.

[29] Ian Foster. 2006. “Globus Toolkit Version 4 : Software for Service-Oriented Systems”,Springer-Verlag. Consultada el 02 de Julio del 2009 <http://www.globus.org/alliance/publications/papers/IFIP-2006.pdf>

[30] Globus Toolkit, http://www.globus.org/toolkit/about.html

[31] Centro Informático Científico de Andalucía. Consultada el 02 dejulio del 2009. <http://www.cica.es/globus-toolkit-2.html>

[32] Sun Grid Engine, http://gridengine.sunsource.net/

[33] Bazán, J. L. , Bolfarine, H. & Branco, D. M. “A generalized skew probit class link for binary regression”. Technical report (RT-MAE-2006-05). Department of Statistics. University of São Paulo, 2006.

[34] Walter R. Gilks, Sylvia Richardson, D. J., “Markov Chain Monte Carlo in Practice”, 1ra. edición. Chapman & Hall/CRC, California 2005.