ULTRACOM: Computación de Alto Rendimiento para Criptoanálisis

Antonio Castro Lechtaler[1,2], Alejandro Repetto[1], Martín Bianchi[1],Marcelo Cipriano[1], Alejandro Arroyo Arzubi[1], César Cicerchia[1], Eduardo Malvacio[1]

[1]EST – Facultad de Ingeniería – Instituto Universitario del Ejército
[2]FCE - Universidad de Buenos Aires
{acastro, marcelocipriano}@iese.edu.ar
{arepetto, mbianchi, cdcicerchia}@ejercito.mil.ar
aarroyo_arzubi@hotmail.com
edumalvacio@gmail.com

Abstract. Los procesos relacionados con la matemática criptográfica se caracterizan por la alta necesidad de cómputo. El objetivo de ULTRACOM es desarrollar una facilidad de computación distribuida multipropósito que permita, como primera aplicación real, el análisis, la validación y la ejecución de pruebas de estrés sobre sistemas criptográficos.
El presente trabajo de investigación detalla el proceso de desarrollo de ULTRACOM y muestra un caso de estudio de su implementación en su versión Beta. Para ello, toma como entrada dos versiones del algoritmo Trivium. Utilizando la misma infraestructura, y sin desarrollo extra, se prueban grandes volúmenes de claves y vectores de inicialización de los algoritmos en busca de secuencias débiles, de menor longitud de la buscada.
Como resultado, se demuestra la capacidad ULTRACOM para poder operar sobre distintos algoritmos sin modificar la plataforma, asegurando la viabilidad técnica para evoluciones futuras.

Palabras Clave: computación de alto rendimiento, computación en grilla de escritorio, criptoanálisis.

1 Introducción

La generación de números primos, fundamentales para la producción de claves, el análisis de secuencias pseudo-aleatorias, los procesos de verificación y validación de algoritmos de seguridad, en un sentido genérico, requieren enormes cantidades de procesamiento. La aplicación inmediata de la computación de alto rendimiento es el ataque por fuerza bruta a algoritmos criptográficos.

Existen sobre la plataforma BOINC algunos proyectos abiertos dedicados a romper los desafíos RSA, así como otros focalizados en encontrar números primos de Mersenne, como el proyecto PrimeGrid. Sin embargo, un problema que encuentra la computación distribuida de este tipo es su alto nivel de especialización. Se comportan como máquinas mono-propósito.

Desde el año 2010, la Facultad de Ingeniería del Ejército está trabajando en la adaptación de la plataforma BOINC a una máquina multi-propósito para que pueda

ser usada con fines de ingeniería en general y de criptografía y criptoanálisis en particular [1]. El objetivo de ULTRACOM es generar una facilidad de computación distribuida multipropósito que permita, como primera aplicación real, el análisis, la validación y la ejecución de pruebas de estrés sobre sistemas criptográficos.

Esta herramienta de evaluación de sistemas criptográficos tiene como fin ser utilizada para la selección y certificación de algoritmos antes de ser incorporados a sistemas de software de uso civil y militar.

2 Computación en Grilla de Escritorio

Las arquitecturas de cómputo distribuidas han sido utilizadas históricamente en problemáticas científicas específicas, como cálculos de flujos, predicciones meteorológicas o cálculos astronómicos. Existen múltiples formas de computación distribuida, desde clústeres de procesamiento, hasta máquinas paralelas, pasando por sistemas de cómputo en grilla. Esta última arquitectura continúa ganando terreno por sobre las otras debido a sus beneficios desde el punto de vista económico.

La computación en grilla fue impulsada por la alta disponibilidad de poder de cómputo ocioso en los equipos, sumado a la mejora continua de anchos de banda de conexión y a los sistemas de estandarización de operación remota, como la tecnología de *web services*, que posibilitan la interconexión de recursos distantes a altas velocidades. El concepto de computación en grilla de alto rendimiento de ejecución remota dio origen a las llamadas grillas de tercera generación.

Las grillas de tercera generación tienen como característica la conformación de super-computadoras a través de la agregación de equipos heterogéneos sobre redes de comunicaciones. Existen distintas clasificaciones para este tipo de tecnología, la más popular las diferencia entre *intergrids* e *intragrids* [2]. Las primeras utilizan los recursos que se encuentran dentro de una organización, sin utilizar vínculos con terceros, como Entropia o Condor. Mientras que las segundas se orientan a captar recursos que exceden a la organización misma, aprovechando al máximo cualquier posibilidad de expandir su poder.

Estas última dieron origen a la computación en grilla de escritorio (*grid desktop comptuing – GDC*). Al salir de los límites de la gestión organizacional, la computación en grilla pasa a convertirse en computación voluntaria. Los que proveen parte (o la totalidad) de los recursos de la grilla no se encuentran bajo la administración de quién controla la grilla, generando algunos desafíos extras.

La GDC, también conocida como computación en grilla voluntaria, se basa en que usuarios externos a la organización donan parte de su poder de cómputo para que se utilizado en pos de resolver un problema dado. Esta lógica de computación colaborativa, conocida *crowd-comptuing* [3], responde a la gran capacidad de cómputo que poseen las computadoras de escritorio. En la mayoría de los casos, los equipos de escritorio se encuentran subutilizados. Estadísticamente, un equipo de escritorio utiliza no más del 20% de su poder de cálculo. Así, la GDC, propone un esquema donde los usuarios donan el 80% del poder restante para resolver problemas de interés para ellos.

La GDC para que sea utilizable, por el contexto en el que se desarrolla, debe tener en cuenta tres puntos principales: la transparencia, la seguridad y la estabilidad.

La transparencia se refiere a que quien solicita la ejecución de un trabajo distribuido no debe saber si el trabajo se está procesando en un gran servidor o en miles de equipos de escritorio.

La estabilidad siempre es importante en los equipos de cómputo. Sin embargo, en la GDC toma mayor importancia considerando que los equipos voluntarios no son controlables fácilmente por el nodo coordinador. Los resultados devueltos por los voluntarios pueden ser incorrectos, ya sea por malicia o por errores en el proceso. Por ello, uno de los componentes claves delos sistemas de GDC son los validadores de resultados.

Por último, el problema de la seguridad no es menor. Para que los voluntarios quieran donar parte de sus equipos a una red de computadoras que ellos no controlan, hay que maximizar los recaudos de autenticación y validación. Las plataformas de GDC usan sistemas de AAA (Autenticación, Autorización y Seguimiento – *Accounting*) comparable con cualquier sistema de redes con infraestructura sobre la Internet.

3 Berkeley Open Infrastructure for Network Computing (BOINC)

Una de las plataformas más conocida en el ámbito de la GDC es BOINC (*Berkeley Open Infrastructure for Nework Computing*). Esta arquitectura fue en la Universidad de Berkeley, California, para dar soporte al proyecto SETI[1] (*Search for ExtraTerriestral Intelligente* – Búsqueda de Inteligencia Extraterrestre), un proyecto que pretende encontrar patrones de señales dentro del ruido blanco captado desde el espacio por una red de radiotelescopios.

BOINC permite que cualquier persona conectada a la Internet descargue un cliente pesado a su equipo y done poder de procesamiento a un proyecto dado. Si bien comenzó con el proyecto SETI, rápidamente surgieron otros proyectos de interés social que llamaron la atención, incluyendo simulaciones realizadas para el LHC, del CERN [4]. En la actualidad, BOINC suma un poder de cómputo de cerca de 9 PetaFLOPS, a través de más de 11 millones equipos donantes [5], una marca interesante si se compara con Tianhe-2, la supercomputadora más veloz del mundo, que opera en el orden de los 33 PetaFLOPS y tuvo un costo de 390 millones de dólares.

Más allá de sus capacidades demostradas, BOINC presenta dos ventajas por sobre otros sistemas de computación de alto rendimiento: bajo costo de adquisición y bajo costo de operación. La adquisición de software y hardware es nula. La plataforma es de código abierto y uso libre, con licencia del tipo GNU Berkeley. El mismo se despliega sobre hardware pre-existente, los voluntarios son los que aportan el hardware. Respecto al costo de operación, la plataforma posee un conjunto de herramientas que permiten la operación de toda la infraestructura de manera fácil. Asimismo, provee una serie de APIs que permite el desarrollo ágil de nuevas aplicaciones.

[1] http://setiathome.berkeley.edu/

Sin embargo su limitante es que los problemas que pueden resolverse son aquellos en los cuales las partes distribuidas son independientes. Es decir, los cómputos para los cuales no es necesario tener una memoria compartida entre los procesos que corren en paralelo. Dentro de la taxonomía de Flynn [6], BOINC está pensado para resolver problemas del tipo SIMD (*Single Instruction, Multiple Data* – Única Instrucción, Múltiples Datos). Más específicamente, BOINC resuelve el problema típico de la ejecución de un conjunto de instrucciones estáticas sobre múltiples entradas de datos, orientado principalmente al rastreo de información en grandes volúmenes de datos, esquema típico que se encuentra en los problemas de criptoanálisis.

Por otro lado, BOINC, como las demás arquitecturas distribuidas, está pensado para ser monopropósito. Para desplegar un sistema que solucione un problema específico, se debe adaptar y ajustar toda la infraestructura.

3.1 Arquitectura de BOINC

La arquitectura de BOINC respeta la arquitectura genérica de todas las plataformas de GDC, que se integra por un gestor de nodos, un gestor de trabajos y un planificador.

El gestor de nodos lleva registro de todos los recursos disponibles en la grilla, con sus capacidades, su nivel de utilización y su confiabilidad.

El gestor de trabajos lleva un registro de los trabajos que la plataforma tiene que ejecutar. Normalmente los gestores de trabajos incorporan lógicas que permiten predecir o estimar las cargas que los trabajos van a necesitar para ser ejecutados.

Por último, el planificador está a cargo de distribuir los trabajos por los nodos, basándose en los requisitos y las disponibilidades. Cuanto más determinístico sea el trabajo, mejor será la planificación. En general la información con la que se cuenta es escasa, por lo que los planificadores operan utilizando métodos heurísticos de optimización.

A estos tres componentes, BOINC les agrega tres más que permiten generalizarla y hacerla más flexible: el *work generator* (WG), el cliente y el *work assimilator* (WA) [7], ver Figura. 1.

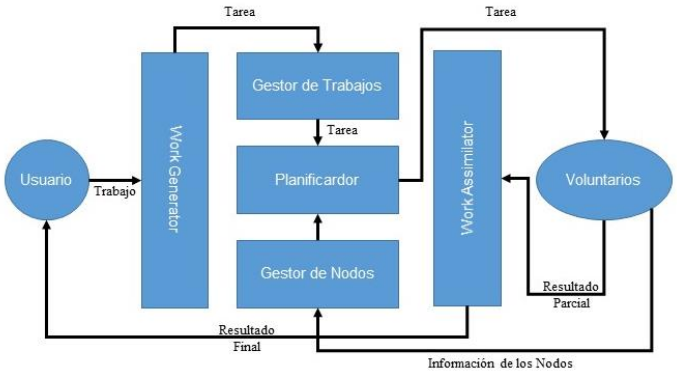

Figura. 1. Arquitectura BOINC

El WG es el componente que posee la inteligencia para dividir los trabajos enviados por el usuario, en tareas, a ser ejecutada por los nodos. Este componente es

parte del conjunto de herramientas que está íntimamente asociada al problema a resolver, pues se debe desarrollar ad-hoc para cada proyecto que se quiera montar sobre la plataforma.

El cliente es la API que permite introducir el código objeto, en el esquema del cliente de BOINC. Cualquier código desarrollado en lenguaje C++ puede ser embebido en el esquema BOINC para que este sea redistribuido a los voluntarios.

Por último, el WA es el componente que contiene la lógica de qué hacer con los resultados parciales. Al dividir el trabajo en subtareas, los resultados parciales deben juntarse de manera de poder obtener un único resultado final. El WA se encarga de recibir los resultados y acomodarlos de manera útil para el cliente.

Esta serie de abstracciones permite que el despliegue y puesta en marcha da un proyecto sobre la plataforma BOINC sea muy simple, como se puede ver [8] y en [9].

Aplicaciones BOINC

Una aplicación BOINC consiste en un programa, una serie de tareas - llamadas *work units* (WU) - y los resultados. Las WU describen la tarea a ser computadas, mientras que los resultados, además de tener la salida de las WU, incorporan datos sobre el rendimiento para poder dar información al planificador para programar tareas futuras.

Un proyecto BOINC puede tener una o varias aplicaciones, que se ejecutan dentro del entorno del cliente que se instala en los donantes. El cliente funciona como si fuera un *sandbox*, un esquema controlado de ejecución aislado que da seguridad al voluntario.

Las aplicaciones residen en una base de datos en el servidor y pueden compilarse para múltiples plataformas. Así, cuando un cliente se conecta anuncia su plataforma y el servidor le envía la última versión disponible.

Work Generator

El WG es un *daemon* de BOINC que se encarga de generar y manejar el trabajo hasta enviar el WU al planificador. En su lógica reside la manera de compartimentar un trabajo en subtareas y la manera de recibir trabajos.

Por cada aplicación debe generarse un WG ya que es el componente que crea los parámetros de entrada que son particulares para cada conjunto de instrucciones. Una vez que se genera el WU, con sus instrucciones asociadas, el WG ejecuta el comando *create_work* para enviar el paquete de trabajo al planificador.

Para la generación de WU, el WG se basa en un archivo llamado wu_template.xml. Este archivo permite definir la cantidad de archivos de entrada, como parámetro, que tendrá la aplicación y cómo serán tratados estos archivos.

De igual modo, se define un result_template.xml que define la cantidad de archivos de salida y el tratamiento que debe dársele a cada uno, incluyendo la facilidad de incorporar una dirección URL para enviarlo automáticamente.

4 ULTRACOM

Trabajos anteriores [10], permitieron generar una capa de abstracción sobre BOINC, en busca de la optimización de la usabilidad, generando una interface Web que

permite crear y administrar proyectos evitando el uso de la línea de comando. Sin embargo, cada proyecto debe ser desarrollado y pensando como un sistema distribuido, yendo en contra del principio de transparencia que persigue la computación de alto rendimiento multipropósito. ULTRACOM propone el rediseño del WG y el cliente de BOINC para poder generalizar la operatoria, mejorando la usabilidad y la transparencia.

La concepción de estos dos componentes parte de la abstracción de los problemas criptográficos más comunes: aquellos que las pruebas se realizan mediante esquemas de fuerza bruta controlada. Este tipo de pruebas se ejecuta mediante la evaluación sistemática de valores de entradas en los algoritmos criptográficos buscando algún resultado particular. En general se pueden probar semillas o valores iniciales en algoritmos de generación de número aleatorios, potenciales claves o pares de números primos que conforman la estructura básica en infraestructuras de clave pública.

La generalización de este problema siempre consta en la iteración controlada de rangos de valores, por lo que el tipo de entrada que esperan recibir los algoritmos presenta una taxonomía genérica del tipo: <cantidad de bits>, <valor inicial>, <valor final>, <paso>. <cantidad de bits> se refiere a la longitud de la entrada esperada, por ejemplo 32 bits; <valor inicial> y <valor final> se refieren a los límites de la prueba; y <paso> indica cada cuantos números probar, en caso de no ser necesario probar por extensión todo el intervalo propuesto.

El archivo ejecutable, también se incorpora como parte de los parámetros de entrada. Como se explicó anteriormente, el cliente de BOINC tiene la capacidad de incorporar un código ejecutable por proyecto. ULTRACOM supera esta limitación haciendo que el código ejecutable del proyecto a su vez ejecute un código externo que se pasa como parámetro. Así, se generando un solo proyecto "ULTRACOM" se pueden ejecutar múltiples códigos a través de una llamada tipo *system*, que permite llamar a un ejecutable dentro a través de otro. El esquema genérico, como lo indica la Figura. 2 sería: el código de ULTRACOM queda embebido dentro del cliente BOINC y este, a su vez, ejecuta las instrucciones del algoritmo a probar con el rango de prueba. Estos dos últimos paquetes de información conforman el *work unit.*

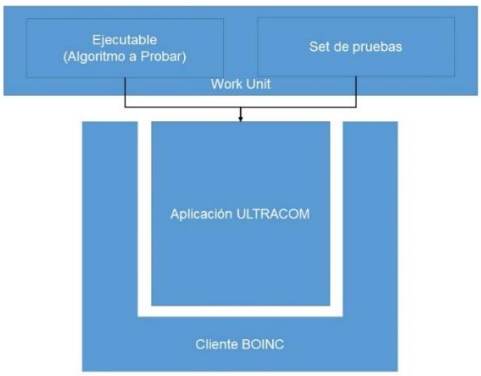

Figura. 2. Esquema de Ejecución de ULTRACOM

Así, el único requerimiento que presenta ULTRACOM es respetar el uso de una API particular que incluye un esquema de entrada y salida para que pueda ser

consumida por la plataforma. El potencial de ULTRACOM se basa en dos principios: la flexibilidad y la escalabilidad.

La flexibilidad de poder ejecutar pruebas de múltiples algoritmos criptográficos en una facilidad de cómputo de alto rendimiento con un esfuerzo igual al esfuerzo del desarrollo del algoritmo en sí, es un valor diferencial. Esta característica mejora la transparencia de la plataforma, haciendo que los analistas criptográficos y desarrolladores no tengan la necesidad de comprender a fondo la plataforma, más allá de la API de desarrollo. El esfuerzo de personalización del WG, el WA y la aplicación que requiere BOINC, se ve facilitada por ULTRACOM, teniendo sólo que desarrollar el algoritmo en cualquier lenguaje de programación, de modo que acepte el formato de entrada y salida propuesto.

La segunda característica, la escalabilidad, es clave en todos los entornos BOINC. Por ser una plataforma voluntaria, se pueden agregar y bajar nodos de manera totalmente transparente. BOINC tiene automatizados los procesos y procedimientos de gestión de nodos. Así, si se suma toda la infraestructura informática de la organización, rápidamente se obtienen poderes de cómputo impensados, aprovechando los recursos ociosos.

5 Caso de Estudio

5.1 Trivium

Trivium es un cifrador de datos sincrónico del tipo *stream cipher* o cifrador de flujo, perteneciente a la familia N-viums. Este tipo de algoritmos permite codificar ráfagas de información bit a bit, haciendo que ante la escucha de un tercero la información aparente ser ruido blanco.

Trivium se basa en un generador de números pseudo-aleatorios que sumado a la información a cifrar arroja como resultado una secuencia en apariencia aleatoria de bits. La fortaleza está en que a través de una clave compartida (semilla), puede generarse exactamente la misma secuencia en el receptor y así poder decodificar la información que se envía sobre la portadora. Este algoritmo, genera secuencia de hasta 2^{64} bit con vectores de inicialización y claves de 80 bits.

La familia N-vium, que incluye Trivium, dispone variaciones en longitudes de claves, de cadenas pseudo-aleatorias creadas y algunas propiedades criptográficas sobre la misma lógica del cifrador de flujo.

Particularmente, en [11], se estudiaron las propiedades de Trivium, Trivium-Toy, Bivium y Bivium-Toy. El resultado destacado de esa investigación fue demostrar que sus versiones Toy (o simplificadas) presentan la misma calidad criptográfica que sus versiones completas, pero con una complejidad computacional menor y, por lo tanto, con una capacidad de producir cadenas de bits pseudo-aleatorios más veloz.

Más allá de este hallazgo, la investigación deja abierta una cuestión clave: matemáticamente se puede comprobar que toda la familia N-vium presenta claves débiles. Es decir, claves para las cuales las secuencias generadas son mucho menores a las previstas y, por lo tanto, muy vulnerables ante ataques.

5.2 Implementación

Aprovechando el conocimiento que se posee sobre la familia Trivium, se decidió realizar la primera implementación de ULTRACOM para evaluar sus distintas versiones, demostrando que la plataforma resiste pruebas consistentes sobre diferentes algoritmos sin mayores modificaciones. Se seleccionaron a modo de prueba el Trivium-32 y el TriviumToy-32.

Si bien el fin principal en esta etapa de la investigación es demostrar la facilidad de uso y la flexibilidad de ULTRACOM, se definió como objetivo de prueba la detección de estados iniciales débiles a través de la ejecución pruebas de fuerza bruta distribuidas y controladas. Así, sin necesidad de ninguna modificación en la infraestructura, se lograron ejecutar pruebas distribuidas sobre dos algoritmos que poseen distintas instrucciones.

5.3 Pruebas y Resultados

Las pruebas se realizaron utilizando dos y tres equipos como infraestructura distribuida. Los equipos voluntarios utilizados son equipos con procesador Pentium ® 4 de 2.8 GHz, con 1.5 GB de memoria RAM y con sistema operativo Microsoft ® 8 de 32bit, enlazados en una red LAN de 100 Mbps. Para el servidor de BOINC se utilizó una máquina virtual sobre un Linux Debian 2.6, con un procesador de 2.8GHz y 512 MB de memoria RAM. Se ejecutaron las mimas pruebas sobre un solo equipo (modo *standalone*) de características similares a los voluntarios para poder comparar la ganancia producida por la distribución del trabajo.

Las dimensiones de prueba fueron la cantidad de equipos y la cantidad de claves probadas (tamaño de la prueba). Sobre cada combinación se ejecutaron tres pruebas para cada combinación de modo de aumentar la fiabilidad de los resultados obtenidos.

Se evaluaron dos parámetros: el tiempo total del trabajo (*turn arround time- TAT*) y la ganancia producida por la plataforma distribuida, conocida como *speed-up* (SU).

Tabla 1. TAT de pruebas distribuidas (en segundos)

Nivel de Distribución	**Millones de claves probadas**				
	50	**150**	**250**	**350**	**450**
Trivium Toy 32					
Standalone	50,4	150,9	249,8	348,2	447,2
2 Voluntarios	124,5	179,2	215,5	265,1	319,5
3 Voluntarios	118,6	156,5	176,4	210,0	244,3
Trivium 32					
Standalone	62,0	187,1	297,3	410,9	509,8
2 Voluntarios	148,2	211,5	265,1	307,5	396,2
3 Voluntarios	137,6	176,8	216,9	243,5	283,3

La Tabla 1 muestra el promedio de los resultados obtenidos, medidos en segundos. Para ambos algoritmos se puede observar que el comportamiento es similar.

Se detecta un punto de equilibrio en el cual la plataforma distribuida demora lo mismo que en la modalidad *stand-alone*. Este es el punto de equilibrio. Para el caso

Trivium Toy 32 este punto ronda los 150 millones de claves testeadas, ver Figura. 3. Debido a la sobrecarga de la distribución, si se desean probar menos de 150 millones, conviene ejecutar la prueba en una sola computadora. Contrariamente, para volúmenes más grandes, se gana tiempo ejecutando la prueba distribuida.

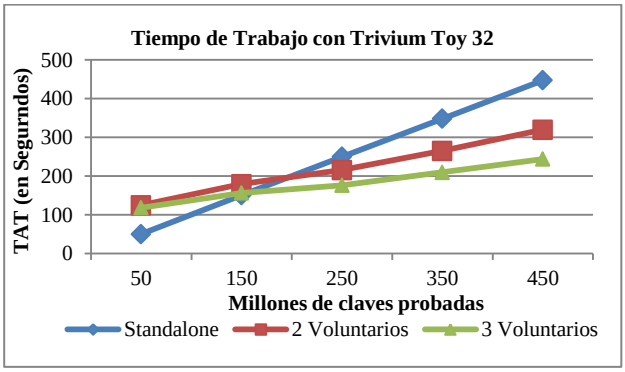

Figura. 3. TAT con Trivium Toy 32

Por otro lado, se puede ver que el SU crece rápidamente a medida que aumenta el tamaño de los trabajos y la cantidad de equipos voluntarios. En el caso de Trivium Toy 32, con sólo tres voluntarios, para evaluar 450 millones de claves, se obtiene un *speed up* de 45%. Vale recordar que esta prueba fue realizada en un ambiente pequeño con el fin de verificar la factibilidad de la realización de pruebas distribuidas en una plataforma multipropósito.

6 Conclusiones

Se implementó una versión *beta* de la plataforma ULTRACOM, comprobando la factibilidad y viabilidad técnica de la generalización de la infraestructura de computación distribuida en grilla BOINC. Este avance, permite extender los horizontes de BOINC y habilita la posibilidad de obtener la capacidad de cómputo de alto rendimiento a bajo costo.

Asimismo, se puso de manifiesto una mejora sustancial de la capacidad de cómputo de la infraestructura distribuida versus la monolítica, inclusive utilizando una red de cómputo.

Así, se concluye que ULTRACOM es un proyecto viable, escalable y de alto impacto para la evaluación y validación de algoritmos criptográficos.

7 Próximos pasos

La versión de ULTRACOM implementada es beta. Esta versión puede ser mejorada en dos dimensiones: cantidad de voluntarios y optimización paramétrica.

Cómo próximos pasos a nivel infraestructura, se desplegará la plataforma en redes de computadoras más amplias, aumentando la cantidad de equipos.

Asimismo, se procederá a evaluar la plataforma en un entorno real, donde los equipos voluntarios no están bajo la administración directa de quién realiza las pruebas, sino que son computadoras de uso cotidiano que aportan su poder de cómputo ocioso a la red. Esto presentará un impacto en el nivel de usabilidad y en la capacidad de cómputo real aportada.

Por otro lado, se concentrarán esfuerzos en la personalización y optimización de parámetros de BOINC que permitan mejorar el rendimiento y la robustez de ULTRACOM. BOINC presenta una cantidad amplia de configuraciones detalladas que, analizadas en detalle y correctamente configuradas, pueden impactar directamente en la capacidad de cómputo de la grilla.

8 Referencias

[1] M. Bianchi, A. Repetto and I. Ariznabarreta Fossati, "Construyendo un Sistema de Cómputo Distribuido Multipropósito," in *Workshop de Investigadores de Ciencias de la Computación*, Paraná, 2013.

[2] L. Ferreira, V. Berstis, J. Armstrong and M. Kendzierski, Introduction to grid computing with Globus, Riverton: IBM Corp, 2003.

[3] D. G. Murray, E. Yoneki, J. Crowcroft and S. Hand, "The case for crowd computing," in *Second ACM SIGCOMM Workshop on Networking, Systems, and Applications on Mobile Handhelds*, New Delhi, India, 2010.

[4] W. Herr, D. Kaltchev, E. McIntosh and F. Schmidt, "Large scale beam-beam simulations for the CERN LHC using distributed computing resources," in *10th European Particle Accelerator Conference*, Edimburgo, 2006.

[5] Boinc Stats, "Boinc Stats," [Online]. Available: http://boincstats.com/en/stats/-1/project/detail. [Accessed 15 07 2014].

[6] M. J. Flynn, "Some Computer Organizations and Their Effectiveness," in *IEEE Transactions on Computers*, 1972.

[7] University of California, CREATING BOINC PROJECTS, Berkeley, 2007.

[8] M. Seil, "Idel Cycles," *Linux Magazine*, vol. 71, 2006.

[9] g. McGilvary, How to Create a BOINC Project, Edinburgo, 2012.

[10] M. Bianchi and A. Repetto, "Computación Distribuida para Seguridad Informática," in *Workshop de Investigadores de Ciencias de la Computación*, 2012.

[11] A. Castro Lechtaler, M. Cipriano, E. García, J. Liporace, A. Maiorano and E. Malvacio, "Model Design for a Reduced Variant of a Trivium Type Stream Cipher," in *XIX Congreso Argentino de Ciencias de la Computación*, Mar del Plata, 2013.

[12] P. Pall, T. Aina, D. A. Stone, P. A. Stott, T. Nozawa, A. Hilberts, D. Lohmann and M. R. Allen, "Anthropogenic greenhouse gas contribution to flood risk in England and Wales in autumn 2000," *Nature*, p. 382–385, 2011.