Berkeley Open Infrastructure for Network Computing (BOINC)

Marina Hernández Bautista[1] y Román Larrosa Lewandowska[1]

[1] Universidad de Granada
marinahb@correo.ugr.es
romanlarrosa@correo.ugr.es

Resumen. BOINC supuso una revolución en el sentido de ser pionero en la utilización del paradigma de la computación voluntaria. Gracias a miles de usuarios voluntarios, multitud de proyectos del área de la medicina, la física, las matemáticas, la astronomía o la química, que necesitan de una gran potencia de computación, pueden llevarse a cabo. Este trabajo hace un recorrido por la arquitectura y las características que hacen de BOINC la red de computadores que forman el sistema distribuido más amplio del mundo.

Palabras clave: computación voluntaria, open source, workunit.

1 Introducción

Berkeley Open Infrastructure for Network Computing o su acrónimo BOINC es un Sistema Distribuido empleado en el ámbito científico para el procesamiento de grandes volúmenes de datos mediante la aglutinación del poder computacional de PCs de particulares.

Se trata de un software de código abierto (Open Source) desarrollado por la Universidad de California en Berkeley, lanzado el 10 de abril de 2002 y cuyo objetivo principal era dar soporte a SETI@home, un proyecto que supuso el primer intento con éxito de computación distribuida, que se requiere para realizar el análisis de las señales radiotelescópicas recogidas para la búsqueda de vida extraterrestre.[1]

2 Descripción

2.1 El paradigma de la computación voluntaria

El paradigma sobre el que BOINC basa su funcionamiento es la computación voluntaria, un acuerdo en el que individuos (voluntarios), ceden recursos de su computador a un determinado proyecto, que usa estos recursos distribuidos para el procesamiento o el almacenamiento de información.[2]

El proceso es el siguiente: el servidor BOINC divide el trabajo en pequeños bloques llamados *workunits*, y éstos son transferidos a los diferentes voluntarios que hayan

instalado el cliente del proyecto en sus dispositivos. Al instalar dicho cliente, se genera un Daemon que se ejecuta al iniciar, y que realizará pequeñas tareas computacionales recibidas por el servidor cuando el dispositivo esté ocioso.[3]

Los resultados son generados en paralelo por los distintos voluntarios y son devueltos al servidor BOINC por la aplicación.

La media de potencia de cálculo en la actualidad es de 24.78 PetaFLOPS, con 135,835 voluntarios que aportan 738,836 computadores.[4]

2.2 Arquitectura

La estructura del servidor de BOINC está formada por cinco daemons, dos de ellos CGI (Common Gateway Interface) [5]:

- **Base de datos MySQL**: contiene las workunits.
- **Feeder**: es un demonio que se encarga de cargar las workunits de la base de datos a una memoria compartida.
- **El scheduler**: es una aplicación CGI que gestiona las peticiones HTTP de los clientes y las respuestas del servidor. Las peticiones normalmente contienen información sobre los resultados obtenidos en los cálculos y sirven también para pedir nuevas tareas al servidor. Las respuestas del servidor son nuevas workunits que los clientes deben ejecutar. Estas unidades son tomadas por el scheduler a través de la memoria compartida.
- **File System**: contiene el código de la aplicación que realiza los cálculos y los datos de entrada para dicha aplicación.
- **Data Server**: es una aplicación CGI a la que accede el cliente para descargar los datos del File System necesarios para procesar la workunit.

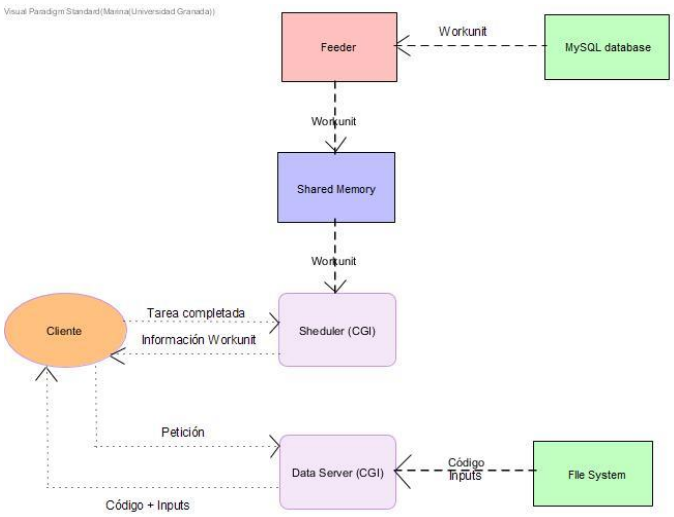

Fig. 1. Arquitectura cliente – servidor de BOINC

2.3 Política de planificación

El cliente de BOINC implementa una política de planificación local, cuyos objetivos son:

- Maximizar el uso de los recursos: el planificador intenta mantener a todos los procesadores del cliente ocupados.
- Satisfacer los plazos de respuesta de los resultados.
- Respetar el porcentaje de recursos dispuesto para cada proyecto por el voluntario.
- Mantener una variedad entre los proyectos a los que el voluntario esta suscrito. El planificador actúa de esta manera para estar siempre ejecutando varios proyectos, ya que un participante puede aburrirse o confundirse si se ha suscrito a varios proyectos y durante mucho tiempo solo ve workunits de uno de ellos. [3]

3 Análisis de características principales

3.1 Compartición de recursos

Como se ha comentado con anterioridad, el cliente solicita las tareas y los datos necesarios a través de dos interfaces: el Data Server y el Scheduler. Estas interfaces son las encargadas de gestionar qué tareas ejecuta cada nodo de cómputo, que en este caso son los computadores voluntarios.

Se trata de un sistema descentralizado ya que el servidor se encarga de dividir una tarea grande en otras más pequeñas que son repartidas a los voluntarios e integradas una vez se calculan sus resultados parciales.

3.2 Sistema abierto

BOINC es un sistema Open Source que cualquiera puede implementar para realizar cálculos de manera distribuida. Todo el código del programa, desarrollado en c++, se encuentra en GitHub [6]. Utiliza un protocolo de comunicación HTTP. Su naturaleza abierta y el uso de protocolos estandarizados permiten la extensibilidad del sistema: añadiendo más equipos conseguimos más potencia de cálculo (hardware), y además se puede dotar al sistema de nuevas funcionalidades (software).

3.3 Concurrencia

BOINC por definición es un sistema que produce paralelismo, ya que los cálculos son ejecutados de manera completamente independiente por los distintos voluntarios, y por tanto con independencia en los recursos que se asignan a cada computador. Cuando un cliente BOINC se comunica con el planificador del servidor y reporta que ha completado un trabajo, recibe las indicaciones de la siguiente tarea. El cliente descarga los

archivos necesarios y maximiza la concurrencia, usando múltiples CPUs cuando es posible y superponiendo computación y comunicación con el servidor. [3]

3.4 Escalabilidad

El servidor de BOINC es muy eficiente, permitiendo millones de respuestas de trabajos al día, pero además tiene una estructura muy escalable, permitiendo incrementar la disponibilidad y capacidad del mismo añadiendo más máquinas. [7]

Además, la arquitectura de comunicación usa *exponential backoff*, un algoritmo que utiliza retroalimentación para estabilizar la tasa de entrada. De esta forma, si el sistema tiene una caída el servidor no sufre ningún colapso de peticiones entrantes. [8]

El backoff exponencial también se extiende a fallos computacionales: si por alguna razón la aplicación falla, el cliente retrasa sus peticiones basándose en el número de fallos para no saturar al servidor.

3.5 Tolerancia a fallos

Los computadores voluntarios no son dedicados al cómputo del proyecto BOINC. Es por ello que el servidor debe tener en cuenta que los intervalos de tiempo de computación de los voluntarios son impredecibles y esporádicos. BOINC monitoriza esos factores y estima el tiempo que se tarda en completar las tareas. Se tiene en cuenta también que los voluntarios están constantemente uniéndose y saliendo del grupo de computadores disponibles, al poder usarse solamente cuando están ociosos. [8]

Además, BOINC soporta computación redundante: un mecanismo para identificar y rechazar resultados erróneos. Esto es implementado mediante los siguientes procesos Daemon:

- **Transitioner:** Implementa la lógica redundante. Genera resultados e identifica condiciones de error.
- **Validater:** Examina conjuntos de resultados y elige el resultado canónico, es decir, el que finalmente se introducirá a la base de datos como resultado correcto.
- **Assimilater:** Gestiona los nuevos resultados canónicos. Da formato al resultado y lo introduce a la base de datos.
- **File deleter:** Elimina ficheros de entrada y salida de los data servers que ya no se van a necesitar.

3.6 Heterogeneidad

Los computadores voluntarios son muy diversos en cuestión de software y de hardware, ya que el cliente de BOINC está disponible para multitud de plataformas (Linux, Windows, MacOS, Android e iOS) con diferentes especificaciones de RAM, CPUs…

El planificador del servidor se encarga de enviar los trabajos a los voluntarios que mejor puedan resolverlos. La dificultad de abstraer al cliente de sus especificaciones hardware podría ser salvada muy pronto gracias al uso de máquinas virtuales en la aplicación cliente. [8]

3.7 Seguridad

Los ejecutables que distribuye un servidor BOINC, son necesariamente firmados digitalmente por un computador offline, previniendo esto de ataques a la base de datos de voluntarios de un servidor para distribuir software malicioso. El proceso de firmado, o *code signing*, está basado en el uso de claves público-privadas. Cada proyecto tiene una clave pública que distribuye a todos las aplicaciones clientes cuando el computador se une como voluntario, y una privada que firma los programas. De esta manera, los clientes pueden determinar si la firma del programa es válida, y solo ejecutan código de programas con firma coincidente.

Cada proyecto mantiene su clave privada en un ordenador que nunca está conectado a la red. Este ordenador es el que se ocupa de firmar el código de los programas que los clientes han de ejecutar. De esta manera se garantiza que los programas son únicamente los que el servidor de proyecto provee, y se eliminan las posibles amenazas. [10]

Para evitar proyectos fraudulentos que puedan acceder a la información de los voluntarios, el cliente de BOINC es ejecutado mediante un usuario sin privilegios en los computadores voluntarios. De este modo, solo puede acceder a la información de los archivos input y a las salidas que produce la propia aplicación, consiguiendo así que no se ponga en peligro la privacidad de los voluntarios.

3.8 Transparencia

Los distintos ficheros de los que un proyecto BOINC hace uso son únicos e inmutables, es decir, no pueden cambiar su nombre. Sin embargo, éstos pueden ser replicados. La descripción de un archivo incluye una lista de URLs de donde puede ser descargado o cargado, aportando transparencia a los clientes, puesto que acceden a cada archivo por su nombre inmutable, pero éste puede ser actualizado o descargado a través de alguna de las réplicas existentes, que pueden ser migradas a distintas localizaciones sin afectar para nada a los clientes. Los servidores y daemons pueden ser igualmente replicados. [3]

4 Conclusión

BOINC es una herramienta muy potente para realizar proyectos que con un bajo presupuesto no podrían realizarse gracias a la participación de todos los voluntarios. Supone una manera muy eficiente de realizar grandes cantidades de cálculos en un tiempo muy y reducido y es por ello que multitud de entidades abren su propio proyecto de BOINC.

Es el caso de Rosetta@HOME, un proyecto de la Universidad de Washington cuyo objetivo es el estudio de las proteínas y la creación de nuevas proteínas sintéticas, así como su visualización en un simulador 3D. Actualmente, los voluntarios del proyecto Rosetta@HOME están contribuyendo a la modelación de importantes proteínas que pueden jugar un factor clave en la lucha contra el coronavirus. [9]

Otro proyecto interesante que se aprovecha de la tecnología de BOINC es Climate-Prediction (climateprediction.net), que investiga aproximaciones de modelos climáticos avanzados. Así, podemos explorar cómo cambiará el clima en el plazo de un siglo debido a pequeños cambios en el modelo, o modificaciones en las cantidades de dióxido de carbono o en el ciclo del azufre.

En definitiva, BOINC hace posible crear y operar proyectos de computación con recursos públicos, soportando infinitas aplicaciones. Las decisiones de diseño de BOINC solucionan muchos de los problemas de la computación distribuida y abren las puertas a un futuro donde la computación voluntaria juega un papel importante en el avance y el desarrollo de multitud de disciplinas.

Referencias

1. SETI@home, https://www.ecured.cu/index.php?title=SETI@home&oldid=1160836, última visita: marzo 5, 2020.
2. Volunteer Computing – BOINC, https://boinc.berkeley.edu/trac/wiki/VolunteerComputing, última visita: marzo 5, 2020.
3. David P. Anderson, BOINC: A System for Public-Resource Computing and Storage, Space Sciences Laboratory, University of California at Berkeley
4. BOINC, https://boinc.berkeley.edu/, última visita: 6 marzo, 2020.
5. G. Farkas, I. C. Szanto, V. Gora and P. Haller, "Extending the BOINC architecture using peer-to-peer application code exchange," 2011 RoEduNet International Conference 10th Edition: Networking in Education and Research, Iasi, 2011, pp. 1-4.
6. BOINC/boinc, https://github.com/BOINC/boinc, última visita: marzo 6, 2020.
7. BoincIntro – BOINC, https://boinc.berkeley.edu/trac/wiki/BoincIntro, última visita: 6 marzo, 2020.
8. Anderson, D. P., Volunteer computing: the ultimate cloud, https://boinc.berkeley.edu/boinc_papers/crossroads.pdf, última visita: 8 marzo, 2020
9. Rosetta@HOME, https://boinc.bakerlab.org/, última visita: 8 marzo 2020.
10. BOINC Security - BOINC, https://boinc.berkeley.edu/wiki/BOINC_Security, última visita: 8 marzo, 2020.