Estudio sobre algoritmos genéticos en la nube y el modelo de programación MapReduce*

G. Muñoz, P. García-Sánchez, P. A. Castillo, M. G. Arenas, A. M. Mora, y J. J. Merelo[1]

Dpto. de Arquitectura y Tecnología de los Computadores, Universidad de Granada
munozs.88@gmail.com,{pgarcia,pedro,mgarenas}@atc.ugr.es,{amorag,jmerelo}@geneura.ugr.es

Resumen Este trabajo presenta el proyecto fin de carrera "Estudio sobre algoritmos genéticos en la nube y el modelo de programación MapReduce". Durante el desarrollo de este proyecto se investigó en el uso y aplicación de Algoritmos Genéticos en distintos entornos de Cloud Computing, como el MapReduce o virtualización de instancias. Se ejecutaron distintas configuraciones de parámetros del algoritmo (como el tamaño de población o el tipo de crossover) en distintas instancias de Amazon Web Services. Los resultados muestran el efecto de estos parámetros al tipo de instancia utilizada.

Palabras clave Cloud Computing, Proyecto Fin de Carrera, Amazon EC2, Virtualización, MapReduce, Algoritmos Evolutivos

Abstract This paper shows the final degree project "A study of genetic algorithms in the cloud and the MapReduce model". During the development of this project the usage and application of genetic algorithms in different Cloud Computing environments was investigated, such as MapReduce or virtualization. Different parameter configurations, such as the population size or crossover type, were launched in different instances of Amazon Web Services. Results show the effect of these parameters to the different types of used instances.

Keywords Cloud Computing, Final Degree Project, Amazon Ec2, Virtualization, MapReduce, EvolutionaryAlgorithms

1. Introducción

En la última década Internet ha transformado nuestra economía y nuestra sociedad. Ha demostrado ser una infraestructura de comunicación y enlace de gran valor que se adapta gradualmente a las necesidades de los usuarios. Con Internet

* Financiado con los proyectos EvOrq (TIC-3903), Beca FPU AP2009-2942, ANYSELF (TIN2011-28627-C04-02) y CANUBE (Proyecto 83 CEI-BIOTIC).

se ha creado una red mundial de intercambio de conocimientos, creatividad y colaboración.

Del mismo modo, Internet ha dado un gran salto adelante gracias al despliegue de la banda ancha de alta velocidad, lo que ha permitido el lanzamiento de muchos nuevos servicios interactivos y de contenido.

Todo esto hace que los programas informáticos ofertados en la red como un servicio disminuyan sus costes y aumenten su eficacia, provocando una gran mejora de la productividad. Desplegado adecuadamente, la llamada Internet del futuro, que cada vez es más presente, trae consigo innovación, aumento de la productividad, nuevos mercados y más crecimiento y empleo en esta década que recién acabamos de comenzar.

En este marco se engloba el Cloud Computing, un nuevo modelo de negocio en Internet, que beneficia tanto al proveedor de servicios en la nube como al usuario, capaz de tener acceso instantáneo a una infraestructura escalable y potente a un coste mínimo.

Paralelamente a esta transformación, la ciencia ha estado evolucionando constantemente durante los últimos lustros, en gran parte, gracias a la aparición y el uso de nuevas tecnologías. La investigación científica avanza al unísono con ésta, pues un cambio tecnológico, la mayoría de las veces, abre un nuevo abanico de posibilidades para la comunidad científica ya sea para el desarrollo en un ámbito clínico, científico o industrial, ya que la necesidad de capacidad de procesamiento de datos es común al sector en el que se desarrollen.

Por otra parte, el crecimiento de los volúmenes de datos a procesar es exponencial, por lo que para resolver un problema complejo el personal científico o investigador tiene que tratar con una cantidad de información ingente. Dentro de estos conjuntos de problemas se encuentran los de búsqueda, a menudo abordados por los denominados Algoritmos Genéticos, que se usan cada vez más en escenarios de gran complejidad.

Sentando estos principios en la sociedad actual, la motivación de este proyecto fin de carrera fué el de buscar una aproximación nueva para desarrollar trabajos de investigación usando las facilidades que nos brinda el Cloud Computing, y concretamente la resolución de problemas de búsqueda representados mediante Algoritmos Genéticos, yendo más allá y sentando las bases de un framework como *Hadoop Map Reduce*, para la reducción de complejidad y tiempo de procesamiento en los métodos de búsqueda.

El resto del trabajo se organiza como sigue: primero se define el concepto de Cloud Computing en la Sección 2. A continuación se explican dos tipos de tecnologías a aplicar los experimentos: *Amazon Web Services* y el modelo *Map-Reduce* y (en las Secciones 3 y 4 , respectivamente). Los Algoritmos genéticos se presentan en la Sección 5. Después se describen los experimentos a realizar (Sección 6) y el análisis de sus resultados (Sección 7). Finalmente se presentan las conclusiones y trabajo futuro.

2. Cloud Computing

Cloud computing es aún un paradigma en evolución, sus definiciones, arquitecturas, modelos, casos de uso, tecnologías base, problemas, riesgos y beneficios, son continuamente redefinidos en debates promocionados por el sector público y privado. En [1] podemos ver varias definiciones aunque muchos citan como más precisa la definición dada por el *National Institute of Standards and Technology* (NIST), donde se define el *Cloud Computing* como:

Cloud Computing es un modelo para habilitar acceso conveniente por demanda a un conjunto compartido de recursos computacionales configurables, por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios, que pueden ser rápidamente proporcionados y liberados con un esfuerzo mínimo de administración o de interacción con el proveedor de servicios. Este modelo de nube promueve la disponibilidad y está compuesto por cinco características esenciales, tres modelos de servicio y cuatro modelos de despliegue.

En un intento de adentrarse en lo que es y no es el Cloud computing compararemos este paradigma con la computación tradicional. Con la computación tradicional ejecutamos copias de software en cada ordenador. Los documentos que creamos son almacenados en el ordenador en el que fueron creados. Aunque se puede acceder a los documentos desde otros ordenadores de la red, no se puede desde ordenadores de fuera de ella.

Con cloud computing, los programas que usamos no se ejecutan en nuestro ordenador personal, sino que son almacenados en servidores accedidos vía Internet. Si tu ordenador se rompiera, el software seguiría estando operativo para los demás. De la misma forma, los documentos que creamos son almacenados en una colección de servidores en los que un usuario con permiso puede, no sólo acceder a los documentos, sino también editar y colaborar en ellos en tiempo real. A diferencia de la computación tradicional, el modelo de Cloud no es PC-Céntrico por lo que el dispositivo que se usa para acceder a la información no es lo más importante.

A continuación, veremos qué es Cloud Computing y qué no es [2] .

2.1. Qué no es cloud computing

Cloud computing no es una red de ordenadores. Con una red, las aplicaciones y documentos son almacenados en un servidor de la empresa y accedidos a través de la red de la empresa. Cloud computing es mucho más que esto. Engloba múltiples empresas, múltiples servidores y múltiples redes. A diferencia de una red de ordenadores, los servicios cloud y el almacenamiento pueden ser accedidos desde cualquier ordenador del mundo con una conexión a internet.

Cloud computing tampoco es la subcontratación de un servicio externo de manera tradicional, donde la compañía contratada presta sus servicios a la empresa contratadora. Mientras una empresa de outsourcing organiza sus datos y aplicaciones, estos documentos y programas solamente son accesibles a los empleados de la empresa a través de la red de la misma, no a todo el mundo vía

internet. Así que, a pesar de las aparentes similitudes, las redes y el outsourcing no son cloud computing.

2.2. Qué es cloud computing

La clave de la definición de Cloud Computing es el Cloud en sí mismo. Para nuestro propósito, el cloud es un gran conjunto de ordenadores interconectados. Estos computadores pueden ser PC's o servidores, públicos o privados.

Esta nube de computadores se extiende más allá de una simple compañía o empresa. Las aplicaciones y datos servidos por la nube están disponibles para un amplio grupo de usuarios, a través de empresas y plataformas. El acceso es vía Internet. Cualquier usuario autorizado puede acceder a estos documentos y aplicaciones desde cualquier ordenador con conexión a Internet. Y, para el usuario, la tecnología e infraestructura que se esconde detrás de la nube es invisible.

Para ejemplificar algunos de los puntos más interesantes del cloud computing nos basaremos en Google, una de las principales compañías impulsoras del Cloud. Según su perspectiva, hay seis propiedades clave en el cloud computing.

- Es "Centrado en el usuario": Una vez que tú eres un usuario conectado a la nube, cualquier cosa que haya almacenada allí - documentos, mensajes, imágenes, aplicaciones - se convierte en tuya. Además, no solo los datos son tuyos sino que puedes compartirlos con otros. En efecto, cualquier dispositivo que acceda a tus datos en la nube también convierte los datos en suyos.
- Es "Centrado en la tarea": En vez de concentrarse en la aplicación y qué hacer, se centra en lo que necesitas y cómo la aplicación puede hacerlo por ti. Las aplicaciones tradicionales - procesadores de textos, hojas de cálculo, email... - se están convirtiendo en más importantes que los documentos que se crean.
- Es poderoso: Conectando cientos o miles de computadores juntos en una nube se crea una riqueza de computación imposible de hacer en un único PC.
- Es accesible: Como los datos son almacenados en la nube, usuarios pueden recuperar información instantáneamente desde múltiples repositorios. No estás limitado a una única fuente de datos, como lo estás en un PC.
- Es inteligente: Con el volumen de datos almacenados en la nube, el data-mining y análisis son necesarios para acceder a esta información de una manera inteligente.
- Es programable: Muchas de las tareas necesarias para el cloud computing deben de ser automatizadas. Por ejemplo, para proteger la identidad de los datos, la información almacenada en un único computador en la nube debe ser replicada en los otros computadores de la nube. Si uno de éstos falla, la programación de la nube redistribuirá automáticamente los datos de éste computador hacia uno nuevo de la nube.

3. Amazon Web Services

En esta sección se presenta *Amazon Web Services* [3] en general, y en particular, los servicios que se han utilizado para el desarrollo de este trabajo.

3.1. Características

Amazon EC2 proporciona un servicio Web que permite obtener y configurar recursos computacionales de forma sencilla. Proporciona un control completo de los recursos computacionales y permite al usuario emplear el entorno de Amazon. Amazon EC2 reduce el tiempo requerido para obtener e iniciar nuevas instancias a minutos, permitiendo escalar la capacidad de forma rápida y en función de los requerimientos. Amazon EC2 permite al usuario adquirir la capacidad que en cada momento necesita. Además, proporciona a los desarrolladores las herramientas para construir aplicaciones robustas y aislarlas de los escenarios comunes propensos a fallos.

Amazon EC2 es un entorno de computación virtual que permite emplear interfaces Web para lanzar instancias de una gran variedad de sistemas operativos, instalar entornos de aplicaciones personalizados, gestionar los permisos de acceso a la red, y ejecutar las imágenes virtuales usando tantos sistemas como sea preciso.

Para emplear Amazon EC2 es necesario:

- Seleccionar una plantilla preconfigurada para ser ejecutada de forma inmediata, o crear una Amazon Machine Image (AMI) que contenga sus aplicaciones, librerías, datos y las opciones de configuración asociadas.
- Configurar la seguridad y el acceso a la red para la instancia de Amazon EC2.
- Escoger el tipo de instancia y el sistema operativo que desee y, a continuación, iniciar, apagar y monitorizar tantas instancias como sea necesario, usando las APIs del servicio Web o la gran variedad de herramientas proporcionadas.
- Determinar si se ejecutarán las máquinas virtuales en múltiples ubicaciones, si se utilizarán direcciones IP estáticas o se conectarán sistemas de almacenamiento persistente a las mismas.
- Pagar sólo por los recursos que se consumen, como el de tiempo de CPU de las instancias o los datos transferidos.

3.2. Aspectos destacados del servicio

- **Elástico**. Amazon EC2 permite incrementar o reducir la capacidad en minutos. Se podría ordenar la iniciación de cientos o miles de instancias de máquinas virtuales de forma simultánea. La aplicación de usuario podría escalar de forma automática los recursos necesarios en función de las necesidades de la aplicación empleando el API del servicio Web de Amazon.
- **Controlado por completo**. Se tiene el control completo de las instancias. El usuario tiene acceso de root a cada una de ellas y, por tanto, podría controlarlas de la misma forma que cualquier otra máquina.

- **Flexible**. Permite seleccionar distintos tipos de instancias, sistemas operativos y paquetes de software. Amazon EC2 permite seleccionar la configuración de memoria, CPU, espacio de almacenamiento, tamaño de la partición de inicio.
- **Diseñado para ser empleado con otros servicios Web de Amazon**. Amazon EC2 emplea el servicio de almacenamiento Amazon Simple Storage Service (Amazon S3), Amazon SimpleDB y Amazon Simple Queue Service (Amazon SQS) para proporcionar una solución completa a la computación, el procesamiento de colas y el almacenamiento en un amplio rango de aplicaciones.
- **Confiable**. Amazon EC2 ofrece un entorno confiable donde puedan reemplazarse las instancias de forma rápida y predecible. El acuerdo de nivel de servicio de Amazon es del 99,95 % para cada una de las zonas de la región de Amazon EC2.
- **Seguro**. Amazon EC2 proporciona numerosos mecanismos de seguridad para los recursos. Incluye interfaces Web para configurar las opciones de firewall entre sus grupos de instancias.
- **Barato**. Se cobra una tasa reducida por la capacidad de computación que se consume. Existen diferentes servicios:
  – *Instancias bajo demanda*. Permiten pagar por la capacidad de computación por cada hora de utilización sin la necesidad de establecer compromisos a largo plazo. Esto permite liberar al usuario de los costes y la complejidad que supone la planificación, la adquisición y el mantenimiento de hardware.
  – *Instancias reservadas*. Las instancias reservadas permiten efectuar un pago inicial bajo, de usa sola vez para cada instancia que desea reservar y disponer de una tasa de descuento significativo en la utilización por hora de la misma.
  – *Instancias Spot*. Permiten al usuario efectuar una oferta por la capacidad no utilizada de Amazon EC2, lo que permitiría ejecutar las instancias durante el tiempo que el precio ofertado supere al precio actual de los recursos no utilizados.

4. Modelo MapReduce

Durante años, los programadores se han visto forzados a realizar implementaciones específicas para trabajar con grandes volúmenes de datos en sus entornos distribuidos de trabajo.

Pese a que no siempre es necesario, el tamaño de la información entrante suele ser grande y los cálculos deben ser distribuidos entre una gran multitud de máquinas para acabar en un tiempo razonable. Los problemas de cómo paralelizar estos cálculos, distribuir la información y manejar los fallos dificultan basatante la labor de implementación.

El modelo de programación MapReduce es una propuesta que pretende resolver las dificultades anteriores, ya que sus características y motivación se basan en la delegación de los cómputos intensivos en datos a un clúster de máquinas remotas que, mediante un sistema de ficheros distribuido, repartirán la carga de

trabajo, optimizando tiempo y recursos. Asimismo, facilita un patrón de desarrollo paralelo para simplificar la implementación de aplicaciones intensivas en datos en entornos distribuidos. Este modelo puede dividir un espacio grande de problema en espacios pequeños y paralelizar la ejecución de tareas más pequeñas en estos sub-espacios.

Este modelo ha cobrado especial interés por su aplicabilidad en entornos de Cloud Computing [2]. MapReduce ofrece unas ventajas muy directas y evidentes como es la centralización de los datos en servidores remotos, eliminando las dependencias con los soportes físicos; o la contratación de servicios en función de las necesidades de las empresas, sin tener que añadir equipos, software o personal, lo que conlleva un considerable ahorro también en el plano energético.

MapReduce [4] es un modelo de programación, desarrollado por Google, que es utilizado para procesar grandes conjuntos de datos distribuidos a lo largo de un clúster de servidores. Este procesado computacional puede tener lugar tanto sobre datos almacenados en sistemas de ficheros, como en bases de datos. El modelo de programación está inspirado en los lenguajes funcionales y permite al desarrollador expresar sus algoritmos utilizando únicamente dos funciones, **map** y **reduce**.

Las funciones map y reduce de MapReduce se definen sobre datos estructurados en pares clave-valor. La función map, escrita por el usuario, recibe un par clave-valor y devuelve un conjunto de pares clave-valor intermedio:

$$map : (k1, v1) \rightarrow [(k2, v2)] \tag{1}$$

Esta función (1) se aplica en paralelo a cada par del conjunto de datos de entrada produciendo una lista de pares $(k2, v2)$ por cada llamada. MapReduce agrupa todos los valores intermedios asociados con la misma clave k y se los pasa a la función reduce.

La función reduce (2) recibe dicha clave y su conjunto de valores asociados y los fusiona para formar un conjunto de valores más pequeño:

$$reduce : (k2, [v2]) \rightarrow [v3] \tag{2}$$

Cada llamada reduce produce bien una lista v3 o un valor vacío. Los resultados de las llamadas se recopilan en la lista de resultados buscada.

Desde la perspectiva del flujo de datos, la ejecución de MapReduce consiste en M tareas map y R tareas reduce independientes. Generalmente, los resultados intermedios se particionan en R trozos para R tareas reduce.

Los principales elementos de un flujo de trabajo MapReduce son los siguientes:

- **Proceso que lee la entrada**: divide los datos de entrada en bloques, siendo asignados cada uno de estos bloques a la función map correspondiente. Estos datos serán leídos de un almacenamiento estable y generará pares clave/valor. Un ejemplo común sería la lectura de un directorio entero, y la devolución de un registro por cada línea de cada fichero.

- **Función de mapeo** (map): Cada función map recibe una serie de pares clave-valor, los procesa individualmente, y devuelve cero o más pares clave-valor de salida. Los tipos de datos de la entrada y la salida de la función map suele ser distintos. Por ejemplo, si la aplicación realizase un conteo de palabras, la función map separaría la línea en palabras y sacaría como resultado la palabra como clave y un 1 como valor.
- **Función de partición**: Las salidas de cada uno de los nodos map son asignadas a un nodo reduce en concreto a partir del resultado obtenido por la función "partition" de la aplicación. Esta función devuelve el índice del reduce buscado, dada una clave y un número de nodo reduce. Una función típica es hallar el valor hash de la clave y hacer módulo del número de nodo reduce.
- **Función de comparación**: La entrada de cada reduce se obtiene de la máquina en la que se ejecutó y ordenó el map utilizando la función de comparación de la aplicación.
- **Función de escritura de salida**: Escribe la salida de la función reduce en un almacenamiento estable, típicamente un sistema de ficheros distribuido.

El principal beneficio de este modelo de programación es la simplicidad. El programador simplemente proporciona una descripción del algoritmo centrada en su funcionalidad.

5. Algoritmos Genéticos

La técnica de búsqueda conocida como Algoritmo Genético se basa en los mecanismos de selección que utiliza la naturaleza, de acuerdo a los cuales los individuos más aptos de una población son los que sobreviven, al adaptarse más fácilmente a los cambios que se producen en su entorno. Hoy en día se sabe que estos cambios se efectúan en los genes de un individuo (unidad básica de codificación de cada uno de los atributos de un ser vivo), y que sus atributos más deseables (es decir, los quele permiten adaptarse mejor a su entorno) se transmiten a sus descendientes cuando éste se reproduce sexualmente.

La primera mención del término Algoritmo Genético, y la primera publicación sobre una aplicación del mismo, se deben a J.D.Bagley [5], que diseñó AGs para buscar conjuntos de parámetros en funciones de evaluación de juegos. Pero es otro científico el considerado creador de los AGs: John Holland, que los desarrolló, junto a sus alumnos y colegas, durante los 60 y 70.

El propósito original de Holland no era diseñar algoritmos para resolver problemas concretos, sino estudiar, de un modo formal, el fenómeno de la adaptación tal y como ocurre en la naturaleza y desarrollar vías de extrapolar esos mecanismos de adaptación natural a los sistemas computacionales.

En [6] Holland presentaba el AG como una abstracción de la evolución biológica, y proporcionaba el entramado teórico para la adaptación bajo el algoritmo genético. El AG de Holland era un método para desplazarse, de una población de cromosomas a una nueva población, utilizando un sistema similar a la selección

natural junto con los operadores de cruce, mutación e inversión inspirados en la genética. En este primitivo algoritmo, cada cromosoma consta de genes (bits) y cada uno de ellos es una muestra de un alelo particular (0 ó 1). El operador de selección escoge entre los cromosomas de la población aquellos con capacidad de reproducción, y entre éstos, los que sean más compatibles producirán más descendencia que el resto. El operador de cruce extrae partes de dos cromosomas, imitando la combinación biológica de dos cromosomas aislados (gametos). Y por último, el operador de mutación se encarga de cambiar, de modo aleatorio, los valores del alelo en algunas localizaciones del cromosoma.

5.1. Algoritmo Genético Básico

En esta sección se describen los componentes de un algoritmo genético básico y sus formas más comunes.

Representación de los Individuos Desde los primeros estudios de los AGs los individuos son cadenas binarias, siendo hoy en día aún la aproximación más utilizada, aunque existen otras que utilizan letras o valores númericos para representar a sus cromosomas.

La sencillez de la representación binaria que utilizan los AGs les aporta características muy importantes de eficiencia. Sin embargo debemos disponer de un método para poder evaluar la adecuación del individuo como solución al problema. Lógicamente, el método de transformación es específico del problema considerado. Sin embargo, a la hora de diseñar el método de codificación es importante tener en cuenta una serie de directrices. Así, debemos buscar una codificación tal que cada punto del espacio de búsqueda esté representado por el mismo número de cadenas binarias, y tal que sea capaz de representar todos los puntos del espacio del problema.

En el contexto de los AGs, el término cromosoma se refiere a un candidato a solución del problema, que frecuentemente es codificado como una cadena de bits. Los genes son tanto un bit como bloques cortos de bits adyacentes que codifican un elemento particular del candidato a solución. Un alelo en una cadena de bits será un 0 o un 1 (para alfabetos largos cada posición puede tener más alelos). El genotipo de un individuo en un AG que emplea cadenas de bits es, simplemente, la configuración de bits del cromosoma de ese individuo.

Cabe resaltar que en este trabajo se usó tanto codificación binaria como codificación real que consiste en exactamente lo mismo pero un gen es representado como un número real, generalmente en un dominio dado por el problema.

Funcionamiento Sea X el problema a resolver, el esquema general de un algoritmo genético es el siguiente [7]:

A continuación se procederá a una explicación exhaustiva de dicho algoritmo. Dada una representación de candidatos a soluciones en una cadena de bits, un AG simple, tal y como se describe en [8], funcionaría del siguiente modo:

1. Comenzar con una población P generada aleatoriamente de n cromosomas de L bits.
2. Calcular el valor de la función de evaluación o fitness (f(x)) para cada cromosoma x de P.
3. Repetir los siguientes pasos hasta que se hayan creado todos los descendientes:
   a) Seleccionar un par de cromosomas de P, siendo la probabilidad de selección proporcional al fitness. Los cromosomas seleccionados serán llamados cromosomas padre.
   b) Con probabilidad p_c (tasa de cruce), cruzar el par de cromosomas padre en un punto (o más) elegido aleatoriamente para formar dos descendientes. Si no tiene lugar ningún cruce, formar dos descendientes que sean copias exactas de sus respectivos padres.
   c) Mutar los dos descendientes en cada lugar con probabilidad p_m (tasa de mutación), y colocar los cromosomas resultantes en la nueva población P'.
4. Reemplazar la población actual P con la nueva población P'.
5. Evaluar la condición de finalización.
6. Volver al paso 2.

En el paso 2 del algoritmo se habla de una función de evaluación, la cual debe ser diseñada para cada problema de manera específica. Dado un cromosoma cualquiera, la función de evaluación le asigna un número real, que se supone refleja el nivel de adaptación al problema del individuo representado por el cromosoma.

Asimismo, la primera operación del paso 3 del algoritmo es la llamada fase de selección. Esta selección se efectúa usando un procedimiento que favorezca a los individuos mejor adaptados. Existen diferentes métodos de selección, que posteriormente veremos.

Cada iteración de este proceso recibe el nombre de generación. Cada generación se obtiene a partir de la anterior por medio de los llamados operadores de reproducción (paso 3 del algoritmo), que pueden ser de dos tipos:

1. **Copia:** Es un tipo de reproducción asexual, en la que un determinado número de individuos pasa directamente a la siguiente generación, sin sufrir ningún proceso de variación en sus genes.
2. **Cruce:** Es una reproducción de tipo sexual en la que se genera una descendencia a partir de un número fijo de individuos, dos por lo general, de la generación anterior. Existen varios tipos de cruce, que veremos más tarde.

El algoritmo acabará cuando se cumpla la condición de fin, que normalmente será una de las siguientes:

1. Se ha encontrado una solución que satisface un criterio mínimo.
2. Se ha llegado a un número determinado de generaciones establecido previamente.
3. Se ha llegado a un límite preestablecido en tiempo de computación.
4. Tras varias generaciones el fitness de la mejor solución no ha variado.

5. Parada manual.
6. Combinación de las anteriores

Posiblemente el criterio más utilizado sea el primero, según el cual De Jong en su tesis doctoral [9] afirmó que si el AG es correcto, la población evolucionará a lo largo de las sucesivas generaciones de tal forma que la evaluación media entre todos los individuos, así como la propia del mejor individuo, convergerán hacia el óptimo global. Se dice que un gen ha convergido cuando al menos el 95 % de los individuos de la población comparten el mismo valor para dicho gen. Se dice que la población converge cuando todos los genes han convergido. Se puede generalizar dicha definición al caso en que al menos un β % de los individuos de la población hayan convergido.

El esqueleto de este algoritmo es la base de la mayoría de las aplicaciones de los algoritmos genéticos. Se podría profundizar mucho más en detalles sobre cuales deben ser las diferentes probabilidades, tamaño de la población y número de generaciones. De esos detalles dependerá, en gran parte, el éxito o fracaso de nuestro algoritmo.

6. Experimentos

Esta sección muestra los experimentos a realizar para hacer comparativas sobre la mejor configuración para AGs en Amazon EC2.

6.1. Parámetros de los algoritmos

Se van a usar diferentes configuraciones para ver el comportamiento y tiempo de ejecución en los distintos entornos. Primero se explicará lo que significa cada columna de las tablas y a continuación se mostrarán.

- **Función**: Función a optimizar, ya sea maximizar o minimizar.
- **Población**: Tamaño de la población de individuos.
- **Individuo**: Longitud de cada individuo en bits.
- **Generaciones**: Número de generaciones que queremos evaluar.
- **Mutrate**: Tasa de mutación en %.
- **Crossrate**: Tasa de cruce en %.
- **Selrate**: Tasa de selección en %.
- **Selección**: Método de selección.
- **Cruce**: Método de cruce.
- **Mutación**: Método de mutación.
- **Reemplazamiento**: Método de reemplazamiento.

Todos los parámetros y configuraciones se han establecido buscando la mayor variedad posible con el objeto de tener diferentes opiniones sobre la ventajas e inconvenientes en escalabilidad, tiempo de ejecución...

Configuración	Población	Bits	Generaciones	Selrate	Crossrate	Mutrate
Config.1	10000	512	100	30	10	10
Config.2	30000	512	200	20	20	20
Config.3	50000	512	500	40	30	10

Cuadro 1. Rastrigin. Tabla de configuraciones establecidas.

Configuración	Selección	Cruce	Mutación	Reemplazamiento
Config.1	Probabilistic (10 %)	One-Point	NRandom (1)	Worst
Config.2	Deterministic	Two-Point	NRandom (1)	Worst
Config.3	Probabilistic (10 %)	One-Point	NRandom (1)	Worst

Cuadro 2. Rastrigin. Metodos de selección, cruce, mutación y reemplazamiento.

Entorno Local La solución secuencial fue probada en una máquina aislada, que corría Ubuntu 12.04 de 64 bits. El equipo en cuestión es un HP Pavilion DV6 3034ss, Intel(R) Core(TM) i5 CPU M 450 @ 2.40GHz, 4 GB RAM.

Entorno Cloud de Amazon La batería de ejecuciones en la nube de Amazon consta de diferentes máquinas virtuales, instancias con diferentes características cada una:

- **Micro**: 613 MB de memoria, un máximo de dos unidades informáticas EC2 (para ráfagas periódicas cortas) y únicamente almacenamiento EBS. Plataforma de 32 o 64 bits.
- **Small**: Dispone de 1.7 GB de memoria, una unidad de computación EC2 (1 núcleo virtual con una unidad informática EC2), 160 GB de almacenamiento, plataforma de 32 o 64 bits.
- **Medium**: Dispone de 3.75 GB de memoria, dos unidades de computación EC2 (1 núcleo virtual con dos unidades informáticas EC2), 410 GB de almacenamiento, plataforma de 32 o 64 bits.
- **Large**: Dispone de 7.5 GB de memoria, cuatro unidades de computación EC2 (2 núcleos virtuales con dos unidades informáticas cada uno de ellos), 850 GB de almacenamiento, plataforma de 64 bits.
- **ExtraLarge**: Dispone de 15 GB de memoria, ocho unidades de computación EC2 (4 cores virtuales con dos unidades informáticas cada uno de ellos), 1690 GB de almacenamiento, plataforma de 64 bits.

Configuración	Población	Individuo	Generaciones	Selrate	Crossrate	Mutrate
Config.1	2500	64	800	40	15	10
Config.2	10000	64	1000	20	30	10
Config.3	50000	64	2000	40	15	10

Cuadro 3. Marea. Tabla de configuraciones establecidas.

Configuración	Selección	Cruce	Mutación	Reemplazamiento
Config.1	Probabilistic (15 %)	Two-Point	NRandom (1)	Generational
Config.2	Deterministic	One-Point	NRandom (2)	Worst
Config.3	Probabilistic (15 %)	Two-Point	NRandom (1)	Generational

Cuadro 4. Marea. Metodos de selección, cruce, mutación y reemplazamiento.

Configuración	Población	Individuo	Generaciones	Selrate	Crossrate	Mutrate
Config.1	10000	512	1000	30	30	10
Config.2	20000	512	3000	50	30	20
Config.3	40000	512	3000	50	30	10

Cuadro 5. OneMax. Tabla de configuraciones establecidas.

- **High-CPU ExtraLarge**: 7 GB de memoria, 20 unidades informáticas EC2 (8 núcleos virtuales con 2,5 unidades informáticas EC2 cada uno), 1.690 GB de almacenamiento de instancias, plataforma de 64 bits.

Los resultados de todas las ejecuciones para cada tipo de instancia, pueden consultarse en la dirección `https://s3.amazonaws.com/secuencial/results/$<$Problema$>$$<$Configuracin$>$.txt`

siendo <Problema> el problema que se desea consultar y <Configuración> el número de configuración. Así, por ejemplo, si se desea consultar los resultados del problema "Marea" con la configuración 2, habría que descargar el fichero desde: `https://s3.amazonaws.com/secuencial/results/Marea2.txt`

Veamos, primero, las tablas de resultados de tiempo de ejecución, para cada problema y tanto en entorno local, como en entorno Cloud, en cada tipo de instancia. Podemos apreciarlos en las tablas

La convergencia de los algoritmos puede ser vista en los ficheros de resultados antes nombrados. No obstante, vamos a estudiar algún caso para verla gráficamente.

Se puede observar que el portatil, el llamado entorno local, se ha comportado muy bien, puesto que tiene tiempos de manera similar a la instancia ExtraLarge y HCPUExtraLarge, por lo que en el caso de querer hacer pocas pruebas, quizás con el entorno local valdría, pero no es el caso, porque hemos hecho una batería relativamente grande de pruebas.

Por otra parte, las instancias Small y Micro, quedarían totalmente descartadas para algún futuro proyecto, puesto que el tiempo excede de los límites permitidos, sobre todo la micro instancia. Estas instancias, pueden ser válidas

Configuración	Selección	Cruce	Mutación	Reemplazamiento
Config.1	Deterministic	Two-Point	NRandom (1)	Worst
Config.2	Probabilistic (15 %)	One-Point	NRandom (2)	Generational
Config.3	Probabilistic (15 %)	One-Point	NRandom (1)	Worst

Cuadro 6. OneMax. Metodos de selección, cruce, mutación y reemplazamiento.

Configuración	Local	Micro	Small	Medium	Large	ExtraLarge	High CPU EL
Config. 1	62	225	117	62	72	58	54
Config. 2	354	Error	703	373	424	338	320
Config. 3	1972	Error	Error	1583	1807	1434	1357

Cuadro 7. Tiempo de Ejecución en segundos. Rastrigin.

Configuración	Local	Micro	Small	Medium	Large	ExtraLarge	High CPU EL
Config. 1	16	47	41	21	24	19	17
Config. 2	164	701	425	222	253	203	169
Config. 3	815	4067	2070	1070	1215	974	818

Cuadro 8. Tiempo de Ejecución en segundos. Marea.

en el Cloud Computing como servidor de pequeñas webs o pruebas que no requieran de demasiado cómputo de datos.

Pasemos ahora a estudiar la escalabilidad, que viendo los ficheros de resultados queda demostrada que existe, pero el objetivo es ver el comportamiento según la configuración y los métodos escogidos. Se ha escogido como instancia representativa, los resultados de la instancia Large, puesto que los tiempos son relativamente buenos y no tiene problemas de algún resultado que no se tiene, por error en la memoria y similares.

Primero, observemos la Figura 1. Esta figura muestra la convergencia al óptimo, que sería 1, durante las 10 primeras iteraciones. Hay que aclarar primero, que la aleatoriedad a la hora de generar la población inicial juega un papel importante pero, aparte de esto, se pueden destacar varios puntos.

- La configuración 2 converge más rápidamente. Esto obedece a que para problemas no demasiado complejos, el reemplazo de los peores se comporta mejor que el reemplazo generacional.
- Las configuraciones 1 y 3 son iguales, excepto en el tamaño de la población y número generaciones. Esto no influye, en la convergencia en las 10 primeras iteraciones y queda reflejado en la figura, que convergen de manera muy similar.
- Otro punto característico de estas dos configuraciones es que la línea de convergencia hay un punto en el que baja en vez de ir en aumento en busca del óptimo. El porqué es muy sencillo, la selección probabilística hace que a veces el mejor sea descartado, que es lo que ha ocurrido en estos dos casos.

Configuración	Local	Micro	Small	Medium	Large	ExtraLarge	High CPU EL
Config. 1	99	583	220	111	134	108	106
Config. 2	664	4043	1280	715	797	634	669
Config. 3	1648	Error	3029	1667	1936	1527	1578

Cuadro 9. Tiempo de Ejecución en segundos. OneMax

Figura 1. Marea. Convergencia en instancia Large para las 3 configuraciones.

Sin nada más que referirnos, pasemos a observar una figura similar, pero correspondiente a la función Rastrigin. En este caso, no hace falta señalar que la convergencia hacia abajo se debe, a que es una función de minimización. En esta figura hay que destacar:

- La configuración 3 es la que se comporta mejor, de manera destacada en las 100 primeras iteraciones de Rastrigin. Por detrás de ella iría la configuración 2 y finalmente la uno parece que es la peor.
- En este caso podemos ver lo anteriormente comentado, parece que la mayor convergencia es fruto del azar y la probabilidad, puesto que las configuraciones 1 y 3 son la misma, pero es bastante probable, que al generar 50 mil individuos saldrá unos mejores individuos que al generar 10 mil como ocurre en la configuración número 1.

Por último, la Figura 3, muestra la convergencia durante 300 iteraciones de la función OneMax, que cuenta los unos de un cromosoma binario, hasta maximizar este número. El óptimo sería 512, que es el tamaño de la cadena elegido para observar los resultados.

- El mejor comportamiento es la configuración 3. Parece que aquí influye de nuevo el número de individuos de la población. Llega al óptimo casi unas 20 iteraciones antes que sus competidores.
- El reemplazo generacional de la configuración 2 vemos que en un inicio de algoritmo puede no ser muy bueno si no hay buena calidad en la población inicial, sin embargo observamos que la convergencia es constante, no como en el caso de la primera, que tras un buen inicio, converge más lentamente hasta ser la última en alcanzar el óptimo. Quizás el hecho de la menor presión de selección haya influido en esto.

Figura 2. Rastrigin. Convergencia en instancia Large para las 3 configuraciones.

7. Análisis de los resultados

Una vez vistos los resultados, analizando las tablas de tiempos, ficheros de resultados y gráficas obtenidas podemos sacar las siguientes conclusiones:

- **Limitación de Algunas instancias**. Algunas de las instancias, como era evidente no tienen memoria suficiente para realizar tareas complejas de cálculos. En nuestro caso, ha quedado demostrado en las instancias micro que no ha podido resolver Rastrigin con la configuración 2 y 3, y la small que no ha podido resolver la configuración 3 del mismo problema, ambas por falta de memoria, debido a que el problema Rastrigin trabaja con un gran número de números reales de precisión doble lo que hace ocupar gran parte de la memoria disponible.
- **Beneficioso**. En los demás casos, la utilización del cloud computing ha sido favorable debido a que los tiempos que nos han dado han sido muy parecidos, en muchos casos mejores, a los de nuestro pc, lo que lleva a pensar que para casos más complejos se podrían comportar mejor.
- **Trabajo Paralelo en Instancias**. Otra ventaja ha sido, que, en el mismo tiempo que se han realizado las ejecuciones para nuestro entorno local, han sido llevadas a cabo todas las ejecuciones en las seis instancias en la nube, ya que, las instancias han trabajado paralelamente y, lógicamente, sin influir unas con otras.
- **Selección**. Para el caso de la selección, en nuestro caso se ha comportado mejor la selección determinística por torneo, puesto que los problemas que hemos tratado no son muy complejos, se ha llegado al óptimo relativamente rápido y no han requerido de una amplía búsqueda ni en la salida de grandes óptimos locales, que son en los casos en los que la probabilística

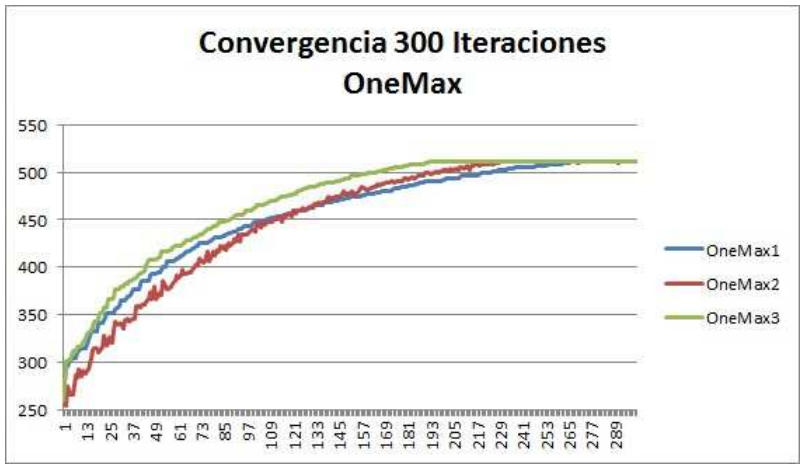

Figura 3. Rastrigin. Convergencia en instancia Large para las 3 configuraciones.

podría resultar favorable. Podemos apreciar una cosa curiosa en la selección probabilística, y es que cuando se ha alcanzado el óptimo, y en todas las iteraciones sea este el mejor individuo, hay alguna iteración en los que se genera uno peor a él y es seleccionado como mejor, quedando una gráfica y tanto rara.

- **Reemplazamiento**. El reemplazamiento generacional depende mucho de las tasas de selección, cruce y mutación elegidas. Si éstas son bajas, la población puede ir incluso decreciendo, pueso que la generación anterior es entera borrada y la nueva contendrá menos individuos. Esto hace que su comportamiento sea ligeramente más rápido en estos casos. Sin embargo, en nuestro problema ha resultado mejor el reemplazamiento de los nuevos individuos generados por los menos adaptados de la generación anterior (Worst), quizás por la misma razón que antes en la selección, no son problemas suficientemente complejos que requieran de una diversidad grande.
- **Cruce**. No se ha encontrado ningún resultado digno de mencionar en cuanto al cruce de uno o dos puntos. Por cuestiones puramente teóricas, podemos decir que el de dos puntos puede dar más diversidad a la población, pero no se ha visto que esto quede plasmado en nuestro estudio.
- **Mejorable**. Para muchos exigentes, esto no es suficiente, necesitan mejoras de tiempos evidentes y por ello se ha propuesto una alternativa mejor con Map Reduce.

8. Conclusiones y trabajo futuro

El Cloud Computing ha llegado con mucha fuerza y todo hace indicar que viene para quedarse. Hemos visto durante, la primera parte que son muchas más

las ventajas que nos ofrece este modelo que los inconvenientes a los que nos podemos enfrentar.

Todo proveedor de computación en la nube, nos ofrece una infraestructura que, a no ser que estemos en una gran empresa o seamos poseedores de superordenadores, no está a nuestro alcance y ello nos hace que instantáneamente podamos acceder a esos recursos y hacer uso de ellos de una forma rápida, sencilla y con un coste bajo. Precisamente esta posibilidad hace que sea una tecnología verdaderamente interesante para el mundo científico e investigador, pues habitualmente en tareas como análisis financiero, meteorológico o bioinformática, entre otras muchas cosas, se trabaja con ingentes cantidades de datos que requieren de millones de cálculos en poco tiempo y precisan de mucho tiempo, pese a tener grandes centros de datos.

Por ello en este proyecto se ha propuesto un pequeño estudio dentro del gran mundo que es la bioinformática, utilizando la infraestructura de Amazon, el mayor proveedor de Cloud Computing y mucho más configurable que Google App Engine, para la ejecución y análisis de algoritmos genéticos básicos.

Se ha implementado una pequeña librería para el diseño de algoritmos genéticos de forma secuencial y un completo manual para la ejecución de estos algoritmos en los servicios web de Amazon. Las características más importantes de esta propuesta han sido:

- **Extensibilidad:** Desde el primer momento de desarrollo se ha pretendido dotar a la librería de un carácter extensible, abstrayendo todo lo posible los distintos elementos dentro de un AG con el uso de interfaces, herencia y demás ventajas de un lenguaje como Java y facilitando el prototipado rápido de AGs mediante la implementación de un número mínimo de funciones.
- **Portabilidad**: La aplicación se encapsula en un fichero JAR que puede ser ejecutado en cualquier entorno anfitrión que disponga de un compilador Java, independientemente de su sistema operativo. Esto se ha hecho así, porque el objetivo principal e era ver las ventajas de computación en la nube, y lo mejor era un sistema portable entre distintas instancias volátiles en Amazon EC2.
- **Sencillez**: El objetivo principal del trabajo, era conocer un poco más distintas posibilidades de servicios de la nube de Amazon y como facilita el análisis de datos y no profundizar mucho en Algoritmos Genéticos, por esta razón se han implementado tres funciones sencillas. El único conocimiento necesario para realizar las pruebas viene explicado detalladamente en el documento adjunto o la documentación generada por cada proyecto.
- **Aprendizaje**: Este trabajo es válido tanto como para alguien quiera iniciarse en el conocimiento de los algoritmos genéticos, como para alguien que quiere sentar las bases de Cloud Computing tener acceso a instancias virtuales en Amazon.

Una vez vistos hecho el análisis de los resultados de la versión secuencial surgió la posibilidad de buscar una alterantiva paralela para un algoritmo genético e investigando se encontró la posibilidad de usar MapReduce, por ello, se ha des-

crito teóricamente en que consiste este modelo y en concreto, la implementación por parte de Hadoop, que es la más conocida.

Para introducirnos en el mundo de Hadoop, se ha explicado como hacer una configuración pseudo-distribuida en nuestro pc para comenzar a escribir aplicaciones de este tipo. Como primer ejemplo de ello, se ha propuesto y explicado la implementación de un básico contador de frecuencias de palabras en un conjunto de archivos, que pese a ser sencillo, nos da una muestra del potencial de MapReduce y de la sencillez de su implementación.

Como extensión se ha presentado una aproximación a un sistema capaz de adaptar la naturaleza intensiva en datos del modelo MapReduce con el carácter iterativo de un Algoritmo Genético, sentando las bases para construir un software que permita ejecutar de forma distribuida algoritmos genéticos paralelos tanto sobre clústers, como sobre el servicio que nos proporciona Amazon Elastic MapReduce.

En líneas generales, se ha descrito el diseño y la implementación de AGs básicos tanto de forma secuencial como paralela utilizando entornos secuenciales y de cloud computing con Amazon como proveedor, y se ha introducido al uso del Framework de Hadoop para el diseño de aplicaciones con MapReduce.

Por último, destacar que las pruebas realizadas con la batería de problemas elegida ("Rastrigin", "OneMAX" y "Marea") arrojan resultados satisfactorios en cuanto a escalabilidad y tiempo, teniendo en cuenta además que nuestra computadora desde la que mandamos los trabajos no consume recurso alguno.

Futuras líneas de trabajo

Hasta el momento solo ha podido desarrollarse un modelo de cómo sería la implementación de un algoritmo genético paralelo usando MapReduce. La principal línea de trabajo sería terminar la implementación de este modelo y hacer extensible las clases implementadas para la parte secuencial a la parte paralela.

Dentro de la línea seguida en el proyecto podemos contemplar otras alternativas que completarían el estudio del modelo de programación MapReduce:

- Se podría investigar si compensa la generación inicial de individuos con una tarea MapReduce como propone [10]. Esto sería útil para utilizar poblaciones de gran tamaño ya que en este caso, tendría un costo inicial muy alto para el nodo maestro mientras los otros se encontrarían ociosos.
- Otro aspecto que podría ser estudiado es la definición de una clase personalizada InputFormat (la encargada de describir la especificación de los datos de entrada de un trabajo MapReduce) para poder configurar el número de tareas map a gusto del usuario.

En lo que engloba al Cloud Computing, hay varias opciones que nos gustaría llevar a cabo:

- Mover una aplicación a la nube, migrar sus datos, configurar una instancia para que no sea volátil, contratando una ip estática y que ésta actúe como servidor, y montar una página web en ella.
- Utilizar proveedores PaaS, como Windows Azure o Google App Engine para desarrollar un servicio en la web.

Referencias

1. Sosinsky, B.: Cloud Computing Bible. First edn. John Wiley & Sons Ltd (2011)
2. Miller, M.: Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online. First edn. Que (2008)
3. Formigacloud: Evaluación de amazon ec2. (2011)
4. J. Dean, S.G.: Mapreduce: Simplified data processing on large clusters. (2008)
5. Bagley, J.: The Behavior of adaptive systems which employ genetic and correlation algorithms. PhD thesis, University of Michigan (1967)
6. Holland, J.: Adaptation in Natural and Artificial Systems. Michigan Press (1975)
7. O. Cordón, L. Magdalena, F.H.: Ten years of genetic fuzzy systems: current framework and new trends. (2003)
8. Mitchell, M.: An introduction to genetic algorithms. (1998)
9. Jong, K.A.D.: An analysis of the behaviour of a class of genetic adaptive systems. PhD thesis, PhD Tesis, University of Michigan (1975)
10. A. Verma, X. Llora, D.G.: Scaling genetic algorithms using mapreduce. (2009)