UNIVERSIDAD DE EXTREMADURA

CENTRO UNIVERSITARIO DE MÉRIDA

INGENIERO EN INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

PROYECTO FIN DE GRADO

INTEGRACIÓN DE UNA HERRAMIENTA DE CÓMPUTO EVOLUTIVO CON UNA DE PROCESAMIENTO MASIVO DE INFORMACIÓN

Autor: Daniel Lanza García

Mérida - Junio 2015

UNIVERSIDAD DE EXTREMADURA

CENTRO UNIVERSITARIO DE MÉRIDA

INGENIERO EN INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

PROYECTO FIN DE GRADO

INTEGRACIÓN DE UNA HERRAMIENTA DE CÓMPUTO EVOLUTIVO CON UNA DE PROCESAMIENTO MASIVO DE INFORMACIÓN

Autor: Daniel Lanza García

Director: Francisco Fernández de Vega

Codirector: Francisco Chávez de la O

Mérida - Junio 2015

Prólogo

En las primeras lineas que describen este proyecto fin de carrera, ...

Agradecimientos

Quiero dar mi mas sincero agradecimiento a A todos muchas gracias.

Índice general

Índice de figuras

Índice de tablas

1. INTRODUCCIÓN

1.1 Motivaciones

La computación evolutiva está adquiriendo cada vez mas importancia a lo largo de los años, sus cada vez mas aplicaciones en sistemas de diferente naturaleza la hacen imprescindible para la resolución de problemas, que haciendo uso de otro modelo computacional, no podrían ser resueltos.

Con el paso de los años y la evolución de los sistemas computacionales, la computación evolutiva se ha intentado aplicar para la resolución de problemas cada vez mas complejos, lo cual en la mayoría de los casos, suele conllevar el uso de más recursos como capacidad de cómputo, memoria y tiempo. Esto hace que se busquen soluciones para mejorar el uso de estos recursos.

Numerosas investigaciones se han llevado a cabo con el fin de minimizar el uso de uno de los recursos mas importantes mencionados anteriormente, el tiempo, se han diseñado algoritmos en este modelo cuya ejecución puede tomar años y que por lo tanto su tiempo de ejecución es impracticable. Varias metodologías se han aplicado para reducir el tiempo de ejecución, una de ellas es la paralelización de parte del proceso, esto puede llevarse a cabo con los procesadores de última generación los cuales poseen varios núcleos de procesamiento o también se puede conseguir con el uso de varios computadores conectados en red.

Este trabajo plantea una solución para utilizar los recursos disponibles de forma eficiente y así reducir los tiempos de ejecución, la solución que se describirá hace uso de una herramienta que explota ambas metodologías, ejecución paralela en procesadores multi-núcleo y uso de numerosas computadoras.

1.2 Objetivos

1.3 Recursos empleados

1.4 Organización del documento

2. ANÁLISIS DEL SISTEMA

En este capítulo nos sumergiremos en los dos campos de conocimiento que este proyecto contempla, la computación evolutiva y el procesamiento masivo de información. Ambos están adquiriendo cada vez más importancia en el mundo de la computación y la resolución de problemas no convencionales.

Se cubrirán temas como porque surgen estás metodologías, sus propósitos y de que manera intentan resolver el problema para el cual han sido desarrolladas.

2.1 Algoritmos evolutivos

Existen problemas computacionales que no pueden ser resueltos con metodologías tradicionales, o porque no existe una solución que pueda proporcionar un resultado aceptable o porque la solución desarrollada necesita de un tiempo y recursos que no son manejables. Para este tipo de problemas se buscan implementaciones que no proporcionan siempre la mejor solución pero que intentar acercarse lo máximo posible, a estos problemas se les conoce como problemas de optimizaron.

Se han desarrollado diferentes formas de afrontar estos problemas de optimización y una de ellas es la computación evolutiva, este modelo se basa en las teoría de la evolución que Charles Darwin postuló. Esta idea de aplicar la teoría Darwiniana de la evolución surgió en los años 50 y desde entonces han surgidos diferentes corrientes de investigación:

- Algoritmos genéticos, desarrolla programas informáticos, tradicionalmente representados en la memoria como estructuras de árboles. Los árboles pueden ser fácilmente evaluados de forma recursiva. Cada nodo del árbol tiene una función como operador y cada nodo terminal tiene un operando.
- Programación evolutiva, una variación de los algoritmos genéticos, donde lo que cambia es la representación de los individuos. En el caso de la Programación evolutiva los individuos son ternas cuyos valores representan estados de un autómata finito.
- Estrategias evolutivas, se diferencia de las demás en que la representación de cada individuo de la población consta de dos tipos de variables: las variables objeto, posibles valores que hacen que la función objetivo alcance el óptimo, y las variables estratégicas, las cuales indican de qué manera las variables objeto son afectadas por la mutación..

Este modelo por tanto, se basa generalmente en la evolución de una población y la lucha por la supervivencia. En su teoría, Darwin dictaminó que durante muchas generaciones, la variación, la selección natural y la herencia dan forma a las especies con el fin de satisfacer las demandas del entorno, de la misma manera pero con el fin de satisfacer una buena solución se basa la computación evolutiva. Podemos observar entonces, algunos elementos importantes como son:

- La población de individuos, donde cada una de ellos representa directa o indirectamente una solución al problema.
- Aptitud de los individuos, atributo que describe cuanto de cerca esta este individuo (solución) de la solución óptima.
- Procedimientos de sección, es la estrategía a seguir para elegir los progenitores de la siguiente

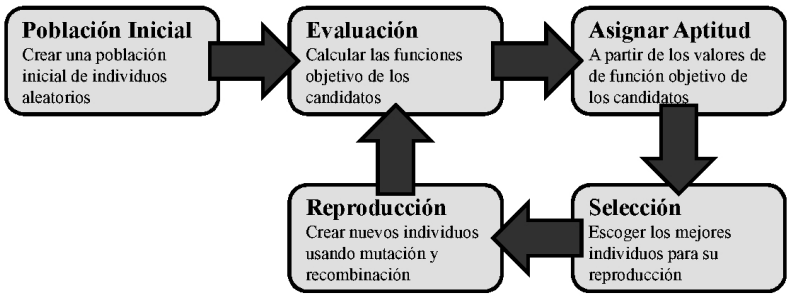

Figura 2.1: Fases del proceso evolutivo

genración. Esta normalmente elige a los individuos mas apto pero existen otras muchas técnicas.

- Procedimiento de transformación, se lleva a cabo sobre los individuos seleccionados y puede consistir en la combinación de varios individuos o en la mutación (cambios normalmente aleatorios en el individuo).

Para llevar a cabo la implementación en computadoras, se ha dividido el problema en diferentes fases y procedimientos, los cuales se ejecutan con un orden determinado, describimos a continuación de forma general como se lleva a cabo la resolución de problemas utilizando este modelo.

1. Inicialización, en esta primera fase se genera la población inicial, normalmente se genera una cantidad de individuos que es configurada y cada uno de ellos se genera de manera aleatoria, siempre respetando las restricciones que el problema imponga a la solución.

2. Evaluación, se calcula la aptitud de cada uno de los individuos de la población para poder determinar posteriormente cuales son más aptos.

3. Las fases que siguen a continuación se repiten hasta que se cumpla una de las siguientes dos condiciones, o que se encuentre la solución óptima o que se alcance un limite impuesto por el programador como un número de generaciones máximo o un tiempo máximo.

   a) Selección, siguiendo la estrategia de selección de progenitores elegida, se eligen individuos de la población. Normalmente los que sean mas aptos.

   b) Procreación, utilizando los individuos seleccionados, se combinan para generar nuevos individuos, y con ellos una nueva población (generación).

   c) Mutación, se eligen normalmente de forma aleatoria individuos a los que se les aplica una modificación también aleatoria, estos nuevos individuos se incluyen también en la nueva población.

   d) Evaluación, todos los individuos de la nueva población son evaluados.

Con el fin de entender mejor este proceso, se muestra una imagen (ver figura 2.1) donde se observan cada una de las etapas descritas anteriormente.

Varias herramientas han surgido en la comunidad para ayudar a la investigación de este modelo, en difierentes lenguajes de programación y plataformas. En nuestro caso hemos elegido ECJ que es un framework bien conocido por la comunidad, implementado en el lenguaje de programación Java y que posee una flexibilidad importante para la ejecución de problemas de muy distinta naturaliza. Más adelante (ver apartado 3.1) se describe con más detalle esta herramienta, explicando su funcionamiento e implementación.

2.1.1 Paralelización

Como hemos comentado anteriormente (ver apartado 1.1) este proyecto surge de la necesidad de optimizar el uso de recursos cuando se intentan resolver problemas complejos con este modelo. Con este fin, y con la posibilidad de paralizar el procesamiento de algunas de las partes del proceso, surge la viabilidad de este proyecto.

Varias partes del proceso evolutivo pueden ser paralizadas, pero no todas merecen el esfuerzo ya que el coste en algunas de ellas es mínimo. Una de las fases que suele conllevar un coste computacional alto y que su paralelización en la mayoría de problemas es sencilla, es la fase de evaluación de individuos.

Se han utilizado diferentes técnicas para este propósito, una de ellas es la ejecución de la evaluación de individuos haciendo uso procesadores multinúcleo/multihilo. Esta téncima consigue buenos resultados pero se limita a las capacidades del procesador que contenga esa computadora. Otro intento para llevar a cabo la paralelización del proceso ha sido la ejecución en diferentes máquinas, las cuales se conectan haciendo uso de una red. Este planteamiento requiere de una implementación más compleja y no suele explotar todos los recursos, ademaás de que algunas soluciones planteadas carecen de la escalabilidad deseada. Tamboén han surgido implementaciones que hacen uso de ambas técnicas, esta solución suele ser la más apropiada ya que hace un uso más eficiente del hardware proporcionado, aunque requiere de una implementación aún más costosa.

2.2 Procesamiento masivo de información

Vivimos en el momento más álgido en la genración de información, nunca antes había existido plataformas que generaran la cantidad de información que se genera hoy. El hecho de que del análisis de grandes cantidades de información se puedan extraer valiosos datos como estrategias de negocio, hacen que numerosas empresas y organizaciones almacenen cantidades ingentes de información para poder sacarle el máximo partido.

La generación de información se está produciendo en ámbitos muy dispares, estos pueden ser redes sociales que mantienen millones de usuarios, grandes empresas con muchos clientes, laboratorios de física con redes de millones sensores y muchos otros ejemplos que podríamos mencionar. Todos ellos queriéndole sacan el máximo valor a la información que recaban.

■ Computación distribuida

Cuando hablamos de cantidades de informaciónón del orden de terabytes o petabytes no podemos pensar en otra cosa que no sea computación distribuida. Para una sola máquina, el tiempo que conllevaría procesar esas cantidades de datos podrían ser del orden de años.

Por tanto, la computación distribuida es la única solución con el hardware que hoy en día manejamos. Esta solución implica el uso de múltiples computadores conectados a la red, cada una de las cuales tiene su propio procesador, arquitectura, etc, con lo que pueden ser totalmente heterogéneas, en contraposición a la computación paralela, que consiste en utilizar más de un hilo de procesamiento simultáneamente para ejecutar un único programa. Idealmente, el procesamiento paralelo permite que un programa se ejecute más rápido, en la práctica, suele ser difiícil dividir un programa de forma que CPU separadas ejecuten diferentes porciones del programa sin ninguna interacción.

2.2.1 Modelo computacional: Map/Reduce

En este ámbito, surge la necesidad de diseñar herramientas que puedan no solo mantener esta información, si no también que tengan la capacidad de analizarla y extraer el valor que se desea de una forma distribuida y con un modelo sencillo.

Google, el buscador de internet más utilizado en el mundo, ha hecho frente ha este problema antes que nadie ya que desde hace años maneja cantidades de información realmente grandes, es por esto que sus investigaciones y experiencia son avanzadas. Varios años atrás hicieron pública [1] una solución para el análisis de grandes cantidad de datos de forma distribuida y con un modelo que da solución a la complejidad de dividir el problema para poder paralelizarlo, ha este modelo se le conoce como Map/Reduce. Esta publicación ha dado pie a que se implemente una herramienta que se conoce con el nombre de Hadoop [3], la cual se describe con más detalle en un capítulo posterior (ver apartado 3.2). La creación de esta herramienta y el hecho de que se hayan obtenidos buenos resultados de ella, ha provocado que surjan otras muchas herramientas a su alrededor, las cuales ayudan a diferentes tareas como el volcado de información (Sqoop), bases de datos para consulta (Impala), gestion de flujos de datos (Flume) y otras muchas.

La propuesta de computación distribuida que se hace en esta publicación, hace uso de un modelo computacional anteriormente conocido en la programación funcional. Este modelo se basa en aplicar dos funciones básicas conocidas como Map (mapeo) y Reduce (reducción). La función de mapeo consiste en aplicar una transformación o procedimiento a todos los datos, obteniendo así la entrada de la siguiente fase, reducción, la cual consiste en "resumir", aplicando la misma función a diferentes partes de la salida de la fase de mapeo, obteniendo finalmente la salida deseada. Numerosas fases de mapeo y reducción pueden ser concatenadas en el orden que se quiera con el fin de producir el resultado esperado. Este modelo es el implementado en Hadoop pero con algunas peculiaridades(ver apartado 3.2.1).

3. DESARROLLO DEL PROYECTO

3.1 Estudio de ECJ, herramienta de cómputo evolutivo

3.2 Estudio de Hadoop, herramienta de procesamiento masivo de información

3.2.1 Implementación del modelo Map/Reduce

3.3 Implementación

4. MANUAL DE USUARIO

4.1 Obtención

El primer paso para la utilización de la solución implementada, no puede ser otro que obtenerla. Con el fin de que la comunidad pueda conseguirla sin problema, se ha creado un repositorio publico desde donde se puede descargar.

El repositorio está localizado en la conocida página por los desarrolladores GitHub (`https://github.com`), en ella se encuentran numerosos proyectos de código abierto y en la cual se ha almacenado este.

Todos los proyectos almacenados en esta página hacen uso de una herramienta de control de versionado llamada Git, la cual hemos también utilizado en este proyecto.

¿Qué es Git?

Es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Hay algunos proyectos de mucha relevancia que ya usan Git, en particular, el grupo de programación del núcleo Linux. Algunas de sus características mas destacadas son:

- Fuerte apoyo al desarrollo no lineal, por ende rapidez en la gestión de ramas y mezclado de diferentes versiones.
- Gestión distribuida. Git le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales.
- Los almacenes de información pueden publicarse por HTTP, FTP, rsync o mediante un protocolo nativo, ya sea a través de una conexión TCP/IP simple o a través de cifrado SSH.
- Los repositorios Subversion y svk se pueden usar directamente con git-svn.
- Gestión eficiente de proyectos grandes.

El proyecto se puede encontrar a través del buscador del sitio buscando "ecj_hadoop", o accediendo directamente al repositorio siguiendo esta dirección: `https://github.com/dlanza1/ecj_hadoop` . Una vez en el repositorio, el proyecto se puede obtener de dos formas, una es pulsando en el botón "Download ZIP"(situado en la columna de la derecha), y una vez descargado descomprimir el fichero, y la otra es clonar el repositorio con Git (crear un repositorio igual de forma local).

Para clonar el repositorio con Git se debe tener instalada esta herramienta en el sistema [4] o utilizar un entorno de desarrollo como Eclipse, el cual la trae incluida.

Para clonarlo desde un entorno de desarrollo como Eclipse [2] el procedimiento suele ser el de importar un proyecto pero con la diferencia de que se debe indicar que la fuente es un repositorio Git, nos solicitará el repositorio que queremos clonar y es ahí donde debemos escribir la URL del repositorio. Algo que debe ser aclarado es que esta operación creará un repositorio local, no genera el proyecto en el entorno de desarrollo, lo cual se explica en la sección siguiente.

Si de otro modo, tenemos la herramienta instalada en el sistema, debemos en primer lugar abrir una consola, posteriormente dirigirnos al directorio donde queremos crear el repositorio local y por último clonar el repositorio con el siguiente comando:

```
[usu@host repo]$ git clone https://github.com/dlanza1/ecj_hadoop
```

Una vez clonado tendremos una copia idéntica del repositorio la cual contiene la última versión del proyecto además de toda la información necesaria para que funcione el sistema de versionado utilizada por Git.

4.2 Importar a entorno de desarrollo

Esta sección tiene como objetivo explicar el procedimiento para importar el proyecto en un entorno de desarrollo, esto no es necesario para su ejecución por lo que no estén interesados en modificar/ampliar el proyecto pueden continuar la lectura en la sección siguiente (Compilación).

En el apartado anterior se ha explicado como obtener el proyecto, pero lo obtenido es básicamente los ficheros de código fuente, no se incluyen proyectos de entornos de desarrollo ni ejecutables.

Si acabamos de descargar/clonar el proyecto, no lo tendremos importado en nuestro entorno, para llevar a cabo esta operación tan solo debemos dirigirnos al menu de importación de proyectos, solucionar que es un proyecto Maven y seleccionar el directorio que contiene el proyecto.

Otra opción, teniendo la herramienta Maven instalada en el sistema [5], es generar primero el proyecto del entorno de desarrollo y posteriormente importarlo como un proyecto normal. Para generar el proyecto del entorno de desarrollo debemos abrir una consola, dirigirnos al directorio donde se sitúa el proyecto y ejecutar el comando Maven correspondiente a nuestro entorno, por ejemplo para Eclipse debemos ejecutar el siguiente comando:

```
[usu@host repo]$ mvn eclipse:eclipse
```

Esto nos generará el proyecto Eclipse y podremos importarlo como cualquier otro proyecto.

4.3 Compilación

Como se mencionaba anteriormente, los ficheros ejecutables no son proporcionados por lo que se deben generar. Con el uso de la herramienta de construcción de proyectos, Maven, explicaremos como construir el proyecto para que pueda ser ejecutado, no obstante, si tenemos el proyecto importado en un IDE, no es necesario utilizar Maven para compilarlo, podemos utilizar los usuales procedimientos para compilarlo.

¿Qué es Maven?

Maven es una herramienta de software para la gestión y construcción de proyectos Java creada por Jason van Zyl, de Sonatype, en 2002. Tiene un modelo de configuración de construcción simple, basado en un formato XML. Es un proyecto de nivel superior de la Apache Software Foundation.

Maven utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado.

El motor incluido en su núcleo puede dinámicamente descargar plugins de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos. Este repositorio pugnan por ser el mecanismo de facto de distribución de aplicaciones en Java. Una caché local de artefactos actúa como la primera fuente para sincronizar la salida de los proyectos a un sistema local.

Maven se ejecuta en unas circunstancias similares a las que explicábamos con Git. Existen dos opciones, o tener Maven instalado en el sistema [5], o utilizar un entorno de desarrollo como Eclipse [2], el cual lo trae incluido.

Si tenemos importado el proyecto en un IDE, para compilarlo (o en nomenclatura de Maven, construirlo) es necesario hacer clic derecho en el fichero incluido en el proyecto, pom.xml, y seleccionar la opción: Run as (Ejecutar como) y en el submenú, Maven build, si hacemos esto por primera vez nos aparecerá un cuadro de dialogo donde se nos solicita los objetivos (goals), ahí debemos indicar "install"(sin las comillas) y aceptar/ejecutar.

En caso de que no estemos utilizando un IDE, necesitaremos tener instalado Maven. Para compilarlo de esto modo debemos abrir una consola, dirigirnos al directorio donde se encuentra el proyecto y ejecutar el siguiente comando:

```
[usu@host repo]$ mvn install
```

Al compilarlo con Maven, tanto desde el IDE como desde la consola, se generará un fichero .jar el cual contiene todas las clases compiladas.

5. TRABAJOS FUTUROS

6. CONCLUSIONES

7. ANEXO I. CÓDIGO FUENTE

BIBLIOGRAFÍA

[1] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Google, Inc.*, 2004.

[2] Entorno de desarrollo Eclipse [online]. URL: `https://eclipse.org/`.

[3] Apache Hadoop [online]. URL: `http://hadoop.apache.org/`.

[4] Procedimientos para la instalación de Git [online]. URL: `http://symfony.es/documentacion/guia-de-instalacion-de-git/`.

[5] Procedimientos para la instalación de Maven [online]. URL: `https://eljaviador.wordpress.com/2013/05/21/guia-de-instalacion-de-maven/`.

[6] Articulo de la Wikipedia sobre Git [online]. URL: `http://es.wikipedia.org/wiki/Git`.

[7] Articulo de la Wikipedia sobre Maven [online]. URL: `http://es.wikipedia.org/wiki/Maven`.