Trabajo de Investigación
Computación de Altas Prestaciones en Proyecto de Integración Android-IoT

Frattini, Maximiliano Gabriel
Rodeiro, Gonzalo
Salva, Ricardo Nicolás
Soro, Emmanuel

[1]Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina

Resumen. El siguiente documento es un trabajo de investigación que aborda la temática computación de altas prestaciones y explica de qué manera podemos implementar los beneficios del paralelismo para mejorar de manera significativa una funcionalidad dentro del proyecto SmartFarm. Dicho proyecto tiene como una de las principales funciones determinar el ritmo de crecimiento de las plantas, en cuanto a su follaje y al diámetro de su tallo, utilizando el procesamiento de imágenes. Actualmente, debido a limitaciones, se procesa solo dos imágenes, por lo que se busca mejorar esto y procesar una gran cantidad de imágenes para obtener datos que se aproximen significativamente más a la realidad. Esto último involucra el procesamiento de grandes volúmenes de datos. Por lo tanto, el principal enfoque de este documento es determinar cómo se puede utilizar la computación de altas prestaciones y el paralelismo para procesar de manera eficiente las imágenes.

Palabras claves: HPC, Paralelismo, Android, JCudaMP, OpenMP, JaMP, Píxel, Espacio de Color, Sistema Embebido, Web Service.

1 Introducción

El trabajo de investigación consiste en el relevamiento y estudio de los paradigmas sobre el uso de paralelismo para aplicaciones en sistemas que requieran altos niveles de cómputo y manejar grandes volúmenes de datos. Se estudiará cuál es el método que más se adapta a nuestras necesidades, y revisaremos el algoritmo que debemos aplicar para el caso propuesto.

En el módulo se va a interiorizar sobre el uso de paralelismo para procesar las múltiples imágenes obtenidas de las plantas para mejorar los datos obtenidos de esta funcionalidad dentro del proyecto anteriormente mencionado. Se brindará una explicación sobre qué herramientas se usan y de qué manera se desarrollan e implementan, adentrando en los algoritmos y software adicional a introducir.

El procesamiento digital de imágenes es una herramienta ampliamente utilizada en la automatización de procesos industriales. La industria agrícola ha comenzado a utilizar este tipo de tecnología para monitorear aspectos de gran relevancia en los cultivos. La productividad de un cultivo en términos biológicos comienza a definirse desde el inicio del ciclo de producción comercial y es afectada por una gran cantidad de factores, algunos propios del genotipo, otros del ambiente y otros de las condiciones de manejo. Para obtener datos confiables, se deben procesar múltiples imágenes de los cultivos, desde variados ángulos, lo que conlleva a un volumen de datos gigantesco. No solo se implementa el procesamiento de imágenes para determinar el crecimiento de los cultivos, sino también para detectar ataques que estos pueden sufrir por plagas cuyo daño puede ser detectado debido a orificios en las hojas de estos.

El desarrollo de algoritmos para procesar imágenes ha crecido de manera sustancial en el ámbito científico, en especial en el campo de la inteligencia artificial para aplicaciones relacionadas con visión por computadora. Este conocimiento se nutre de la digitalización de datos de dispositivos para captar imágenes. Por tanto, las imágenes son el núcleo para poder describir la realidad.

El procesamiento de imágenes se usa en una gran variedad de sistemas. Un ejemplo claro que se puede encontrar en la mayoría de los dispositivos celulares de última generación es el reconocimiento facial. Otra implementación, que aun se encuentra en desarrollo, pero con grandes avances, es el automóvil desarrollado por Tesla, cuyo objetivo es lograr el piloto automático. Utiliza 8 cámaras, que ofrecen una visión de 360 grados alrededor del vehículo con un alcance de 250 metros. Implementa el procesamiento de las imágenes en tiempo real obtenidas por las cámaras, más información obtenida con otros dispositivos, como, por ejemplo, sensores ultrasónicos. Otro sistema es Plagapp, que utiliza el procesamiento de imágenes con el objetivo de que agricultores puedan reconocer la plaga que está afectando un cultivo y, así, poder determinar las acciones necesarias para eliminarla. Implementa software de código abierto para trabajar con imágenes, como OpenCV y Matlab.

Hay dos conceptos importantes de las imágenes: el espacio de color y el píxel. El espacio de color más utilizado para procesar imágenes es el RGB (Red – Green – Blue), aunque también se encuentra el espacio HSV o HSB (Hue-Saturation-Value) que es una transformación no lineal del espacio de color RGB para medir las progresiones de color. La elección de un espacio de color está en

función del uso que se le desea dar a la imagen. El píxel es la unidad básica de una imagen. Una imagen consiste en un conjunto de pixeles. Entonces, se puede pensar a la imagen como una gran matriz, donde cada celda es un píxel y se puede determinar su "color" o "intensidad". Por tanto, a modo de dar un ejemplo, una imagen con una resolución de 500x300 significa que está conformada por una matriz de pixeles con 500 filas y 300 columnas. Se los puede representar en color y en escala de grises. En una imagen en escala de grises, cada píxel tendrá un valor entero dentro del rango [0-255], donde 0 corresponde al color negro, 255 al color blanco y valores entre ellos representan variaciones de los tonos de gris. Como se mencionó anteriormente, el espacio de color más usado es el RGB, por tanto, los pixeles de estas imágenes estarán conformados por el vector RGB. En el vector, cada uno de los colores estará representado por un numero entero dentro del rango [0-255]. Así, un píxel cuyo color sea rojo estará representado por el vector (255,0,0), uno blanco será (255,255,255), uno negro (0,0,0), uno amarillo (255,255,0), etc. [1]

Dado que las imágenes tienen muchos datos por sí solas, hay algunos que no son relevantes, por lo que, teniendo en cuenta la gran cantidad de datos que se manejan, es necesario eliminar aquellos que no sean de importancia. Para lograr esto existen filtros de segmentación de imagen, como los filtros morfológicos. Estos filtros simplifican los datos de las imágenes preservando sus formas características esenciales y eliminando las irrelevantes. También existe el difuminado gaussiano cuyo objetivo es suavizar la imagen con una función Gaussiana. Se usa para reducir el ruido digital y los detalles en las imágenes. Así, se logra eliminar el ruido digital evitando perder información importante. [2]

Recientemente, la velocidad computacional se ha incrementado enormemente. Por ese motivo, para lograr procesar todas las imágenes dentro de tiempos aceptables investigamos acerca de las herramientas disponibles en el entorno de programación Java en donde podamos escribir programas multihilos y aprovechar todo el poder del paralelismo. Para empezar, uno de los principales inconvenientes a la hora de usar paralelismo es la dificultad de programación al hacer uso de thread & runnable para desarrollar aplicaciones, ya que se debe lidiar manualmente con problemas de partición, sincronización de datos e hilos, tanto como el equilibrio de carga.

Por tanto, se propone no solo lograr la paralelización del procesamiento, sino también aprovechar las prestaciones que pueden brindarnos GPGPU. Para esto utilizaremos el framework JCudaMP[3] que incorpora JaMP, que es una adaptación de OpenMP para el lenguaje de programación Java [4]. JaMP soporta OpenMP 2.0 y parcialmente las nuevas funciones de OpenMP 3.0. JaMP genera código puro de java que puede correr en cualquier JVM. Además, el framework es capaz de analizar dinámicamente el entorno de ejecución para detectar el hardware y así, en base a los resultados, reescribe el bytecode y genera el código GPGPU necesario para poder aprovechar un alto grado de paralelismo.

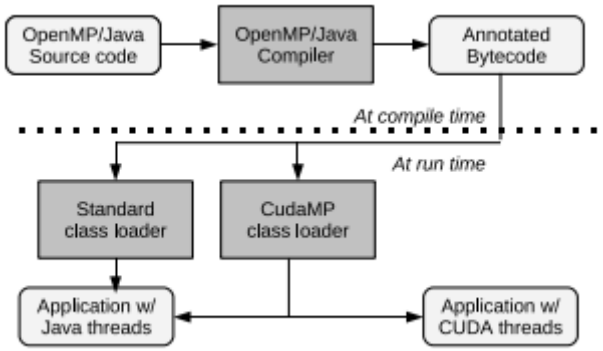
Las distintas directivas OpenMP se incorporan al código como comentarios.

También es capaz de traducir *parallel for* para que pueda correr en tarjetas gráficas habilitadas para CUDA para ganar más velocidad de procesamiento. Si el bucle no es compatible con CUDA, lo traduce a una versión *threaded (enhebrada)* que usa los núcleos de una maquina multi-núcleo típica.

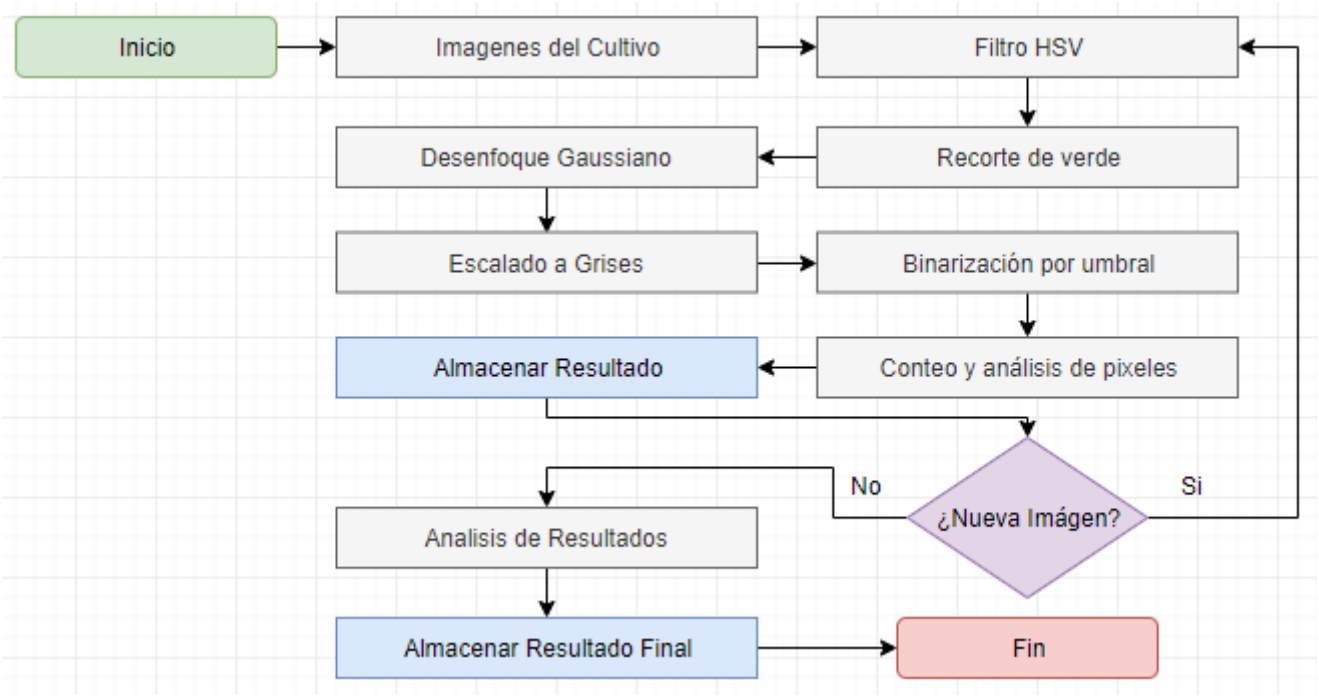
[1] https://revistas.ulima.edu.pe/index.php/Interfases/article/view/1767/1819
[2] https://revistas.unal.edu.co/index.php/acta_agronomica/article/view/42657
[3]researchgate.net/publication/228340673_JCudaMP_OpenMPJava_on_CUDA
[4] https://www2.informatik.unierlangen.de/teaching/thesis/download/i2S00425.pdf

2 Desarrollo

Para el desarrollo del software utilizaremos el entorno de desarrollo JCudaMP. A continuacion, se muestra el modelo de ejecucion de programas OpenMP/Java en entorno CUDA.



Para implementar un procesamiento de imágenes para medir el ritmo de crecimiento de cultivos, se requieren de un gran volumen de imágenes, desde distinto angulos, para poder compararlas y asi obtener datos precisos. Como se explico anteriormente, este volumen de datos procesado de manera secuencial le llevaria al sistema un tiempo significativo, para poder suplir esto se propone utilizar OpenCV para procesar las imágenes y las prestaciones brindadas por el framework JCudaMP para lograr paralelizar altamente el procesamiento de las mismas.



Esquema general para implementar un Sistema de procesamiento de imágenes de cultivos.

En el esquema anterior se ve, de manera clara, que el procesamiento de las distintas imágenes se puede paralelizar. Para desarrollar el algoritmo nos basamos en el paper *Procesamiento de Imágenes de plantas ornamentales multiescala para calcular su procesamiento*. En el documento se explican las dificultades que conlleva el procesamiento de imágenes de los cultivos, los diversos factores que provocan que las imágenes difieran en el tiempo, no solo por el crecimiento del cultivo sino también por factores climáticos, como el viento. Por esto, se da la necesidad de tomar una gran cantidad imágenes desde múltiples ángulos (al menos 3) y luego de procesarlas, cruzar los datos obtenidos para poder llegar a poder establecer de manera correcta el ritmo de crecimiento del cultivo. Este procesamiento será realizado en un web service de la aplicación, que entregará los datos finales a la aplicación de Android, donde el usuario podrá ver el ritmo de crecimiento histórico de los cultivos mediante gráficos. De esta manera, dejamos fuera de la aplicación Android el arduo trabajo de almacenar las imágenes y procesarlas.

El web service recibirá las imágenes e ira procesándolas en paralelo, a medida que las recibe, y al finalizar todas las imágenes, cruzará los datos para, así, obtener información que se aproxime lo máximo posible a la realidad del cultivo. Por último, se almacenará el dato correspondiente al cultivo para cuando haya una consulta de este, realizada por la aplicación Android.

3 Explicación del algoritmo.

A continuación, se detalla un pseudocódigo de cómo se podría implementar, de manera global, los métodos anteriormente mencionados. Se introduce en el código Java nuevos directivas openMP.

```
public static void main{

        Global List<Image> imglist = new List<>(); //Lista que tendrá las imágenes

        new double data; //Almacenara el resultado final

        capturarImagenes(); //Hilo que agrega imagenes recibidas de la camara IP

        Global double img_vec_data[] = new double[imglist.size]();

        //#pragma omp parallel for nowait

        for(int i=0; i<imglist.size; i++){

//Procesa la imagen en la pos i de la lista y guarda el resultado en la pos i de img_vec_data

                procesarImagen(i);

        }

//Cruza los datos que se encuentran en img_vec_data y devuelve un único resultado que representara el nivel de follaje actual del cultivo.

        data = cruzarDatos();

//Almacena el dato en la BD para posteriormente poder realizar graficos historicos

        guardarData(data);

}
```

En el código utilizamos la potencia que nos puede proporcionar GPGPU para procesar de manera altamente paralela todas las imágenes recibidas con la directiva *parallel* de OpenMP. Incorporamos la cláusula “nowait” debido a que no es necesario que se espere a determinado hilo de un bloque de hilos para que otro hilo que procese una imagen pueda lanzarse. Para procesar las imágenes utilizamos la librería OpenCV, originalmente desarrollada por Intel, que incluye métodos muy eficientes.

Explicación de funciones:

capturarImagenes(): Esta función despliega un thread asincrónico encargado de establecer una comunicación con el sistema embebido y la cámara IP, y de esta forma traer las imágenes correspondientes.

procesarImagen(int pos): Procesa la imagen correspondiente a la posición "*pos"* de la lista y almacena el resultado en la posición "*pos"* de img_vec_data.

cruzarDatos(): Realiza la cruza de todos los datos contenidos en img_vec_data y, así, retorna el resultado.

guardarData(double data): Almacena el resultado final del procesamiento en la BD.

4 Pruebas que pueden realizarse

Para realizar esta prueba, se debe instruir al sistema embebido desde el web service que se realizará la toma de imágenes. El embebido preparará el entorno lumínico de la planta para que las imágenes sean las correctas y nos comunicara que todo se encuentra listo. Entonces, el web service enviara peticiones a las distintas cámaras para que tomen las imágenes y las retornen. Así, en el servidor recibiremos todas las imágenes tomadas y procederemos a procesarlas, de manera paralela, gracias al entorno de desarrollo JCudaMP, pudiendo aprovechar las prestaciones de la GPU. Una vez finalizado el procesamiento, se realizará la cruza de los datos y se almacenará el resultado final en la BD.

Los casos de prueba que debemos realizar son, variar la cantidad de entradas de imágenes para poder determinar que tan eficiente es el algoritmo. Además, podemos probar al sistema en un entorno sin GPU ya que JCudaMP también permite aplicar las directivas OpenMP en CPUs con múltiples núcleos.

5 Conclusiones

El presente trabajo da una breve introducción de cómo se podría aprovechar distintas tecnologías de procesamiento paralelo para integrar a nuestro presente proyecto. Estas tecnologías (GPU, OpenMP, JCudaMP) son optimas ya que pueden funcionar en un entorno de trabajo con limitaciones, no requieren un costoso hardware como un supercomputador. Por esto, se propuso una alternativa para procesar de manera paralela grandes volúmenes de datos en un pequeño servidor.

Presentamos el entorno de desarrollo JCudaMP, que esta al alcance de cualquier desarrollador que quiera aventurarse en el mundo de la GPGPU. Este framework es relativamente nuevo, pero el uso de las GPU como GPGPU ya es algo que se da hace unos años. Así, se puede obtener un gran beneficio de su poder de procesamiento altamente paralelo. Por tanto, notamos que explotar este tipo de herramientas provee un gran beneficio.

6 Referencias

1. Georg Dotzler, Ronald Veldema, Michael Klemm, University of Erlangen-Nuremberg – Computer Science Department 2, Germany: JCudaMP: OpenMP/Java on CUDA.
2. Césares Farías, C.A., Farías Mendoza, N., García Díaz, N. y García Rebolledo, A. (2017). Procesamiento de imágenes de plantas ornamentales multi-escala para calcular su crecimiento. 3C TIC: Cuadernos de desarrollo aplicados a las TIC, 6(3), 10-25.
3. Andres Fernando Jimenez Lopez, Marla Carolina Prieto Pelayo, Ángela Ramírez Forero: Enseñanza del Procesamiento de Imágenes en Ingeniería usando Python. VAEP-RITA Vol. 3, Núm 4, dic. 2015.
4. Petr B˘elohl´avek: OpenMP for Java. Department of Distributed and Dependable Systems. Prague 2015.
5. José Antonio Taquía Guitiérrez: El procesamiento de imágenes y su potencial aplicación en empresas con estrategia digital. Universidad de Lima, Colombia. 15 de Octubre del 2017.