UTILIZACIÓN DE LA PLATAFORMA HADOOP PARA IMPLEMENTAR UN PROGRAMA DISTRIBUIDO QUE PERMITA ENCONTRAR LAS PAREDES DE CÉLULAS DE LA EPIDERMIS DE PLANTAS MODIFICADAS GENÉTICAMENTE

GUSTAVO IRVING CALI MENA
ERICK ROGGER MORENO QUINDE

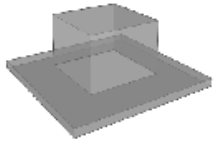
AGENDA

- Introducción
  - Antecedentes
  - El problema
  - Objetivos
- Algoritmo de detección de paredes de las células.
- Hadoop (MapReduce)
- Amazon Web Services
- Diseño e Implementación
- Pruebas y Resultados
- Conclusiones y Recomendaciones

ANTECEDENTES (I)

- Se tiene un conjunto de imágenes DIC de las células de plantas del tipo Arabidopsis Thaiana

Objeto transparente

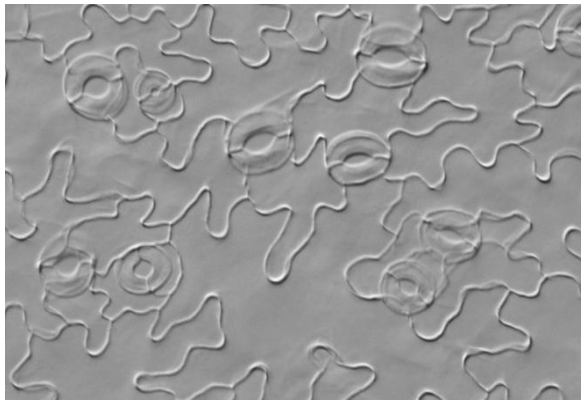


Desfase o polarización

Interferencia

- Existe un algoritmo que encuentra las paredes de las células usando este tipo de imágenes implementado en Matlab.

Ejemplo de Imagen DIC



ANTECEDENTES (II)

- El procesamiento digital de imágenes en general es costoso tanto en tiempo como en recursos (hardware).
- Una de las librerías de mayor acogida actualmente para procesamiento digital de imágenes es OpenCV.
- Hadoop fue concebido originalmente para el procesamiento de texto.

OpenCV

hadoop

EL PROBLEMA

- Se necesita procesar una gran cantidad de imágenes en el menor tiempo posible.
- Las imágenes capturadas tienen una resolución de 2560 X 1920 píxeles.
- El algoritmo implementado en Matlab es lento.

OBJETIVOS

1. Re codificar el algoritmo de Matlab a una aplicación en C++.

2. Adaptar el código C++ que obedezca el paradigma MapReduce para poder ser ejecutado en Hadoop.

OBJETIVOS (II)

3. Adaptar las clases de entrada y salida de Hadoop para el manejo de imágenes.

4. Realizar la comparación de los tiempos de procesamiento entre los 2 lenguajes en que se implementó la solución.

ALGORITMO DE DETECCIÓN DE PAREDES DE LAS CÉLULAS (I)

El Algoritmo de procesamiento para la detección de las paredes celulares que se implementó se divide en 5 pasos:

1.- Filtros de coincidencias.

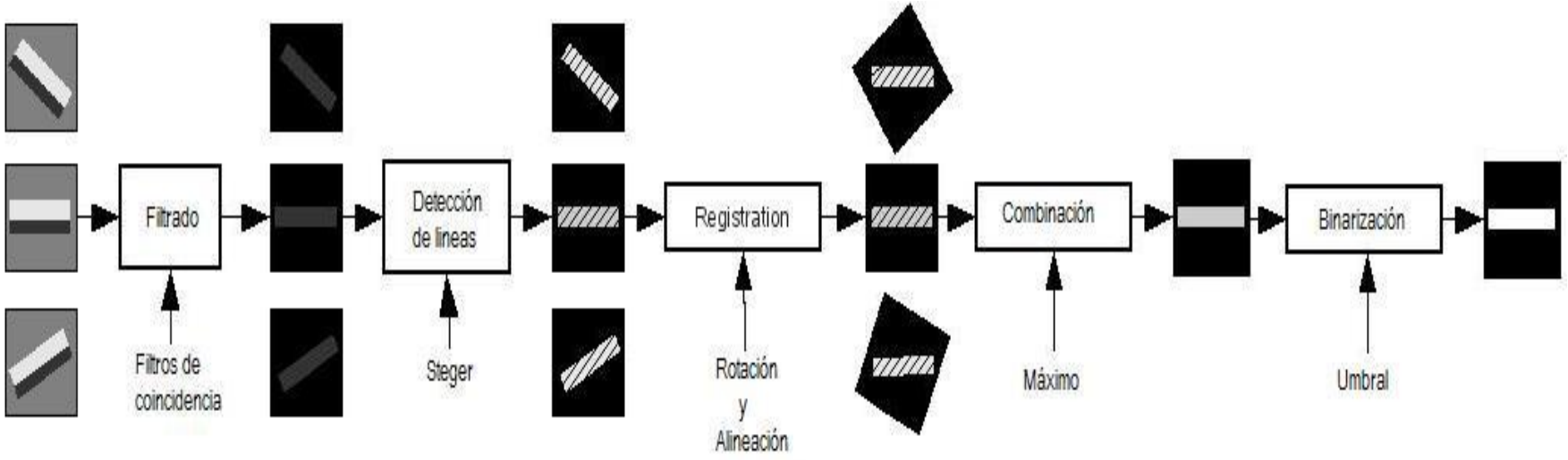
2.- Detección de las estructuras curvilíneas.

3.- Image Registration.

4.- Combinación de imágenes.

5.- Binarización de la imagen.

ALGORITMO DE DETECCIÓN DE PAREDES DE LAS CÉLULAS (II)



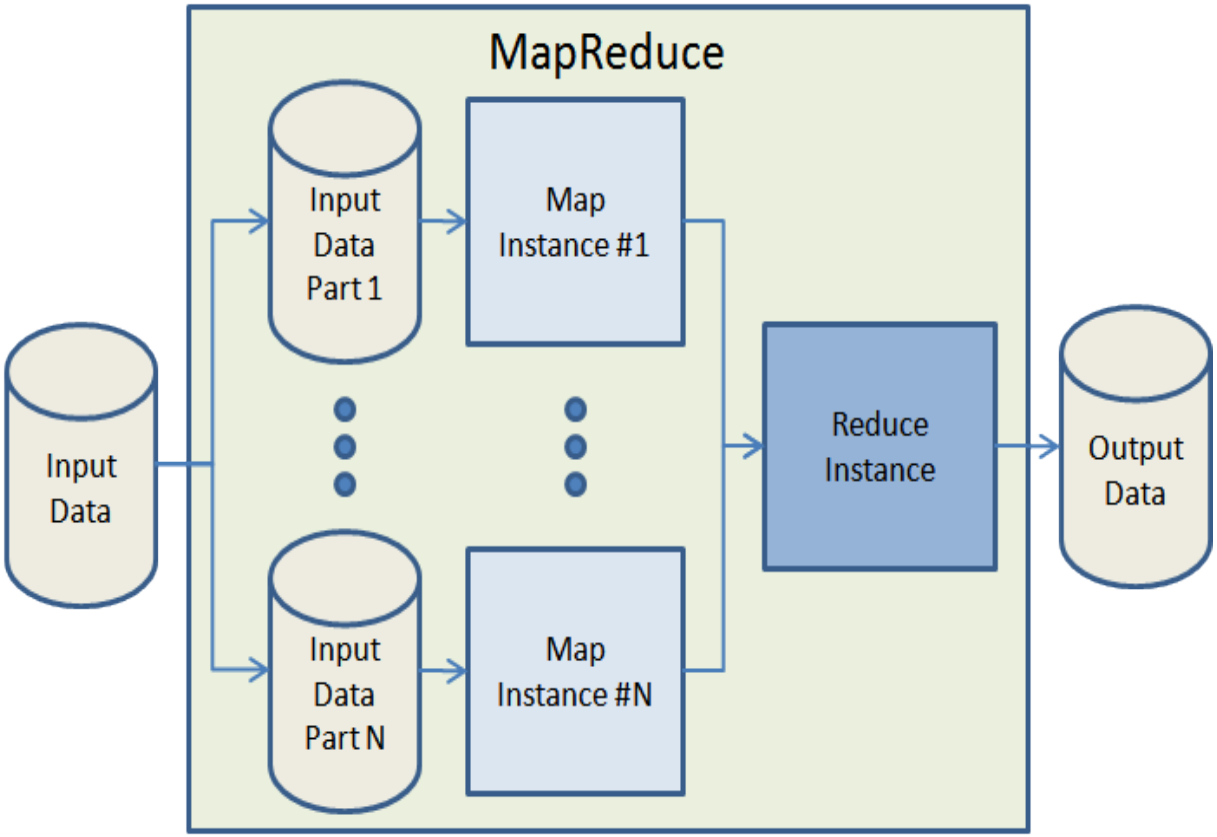
HADOOP

Es un framework open-source desarrollado en Java que nació como proyecto de Apache para el procesamiento y consulta de grandes cantidades de datos sobre un grupo de computadores.

La programación en Hadoop implementa el modelo de programación MapReduce desarrollado por Google.

Hadoop Pipes, es una solución a la ejecución de aplicaciones desarrolladas en el lenguaje C++.

MAP REDUCE


MapReduce
Input Data
Input Data Part 1
Input Data Part N
Map Instance #1
Map Instance #N
Reduce Instance
Output Data

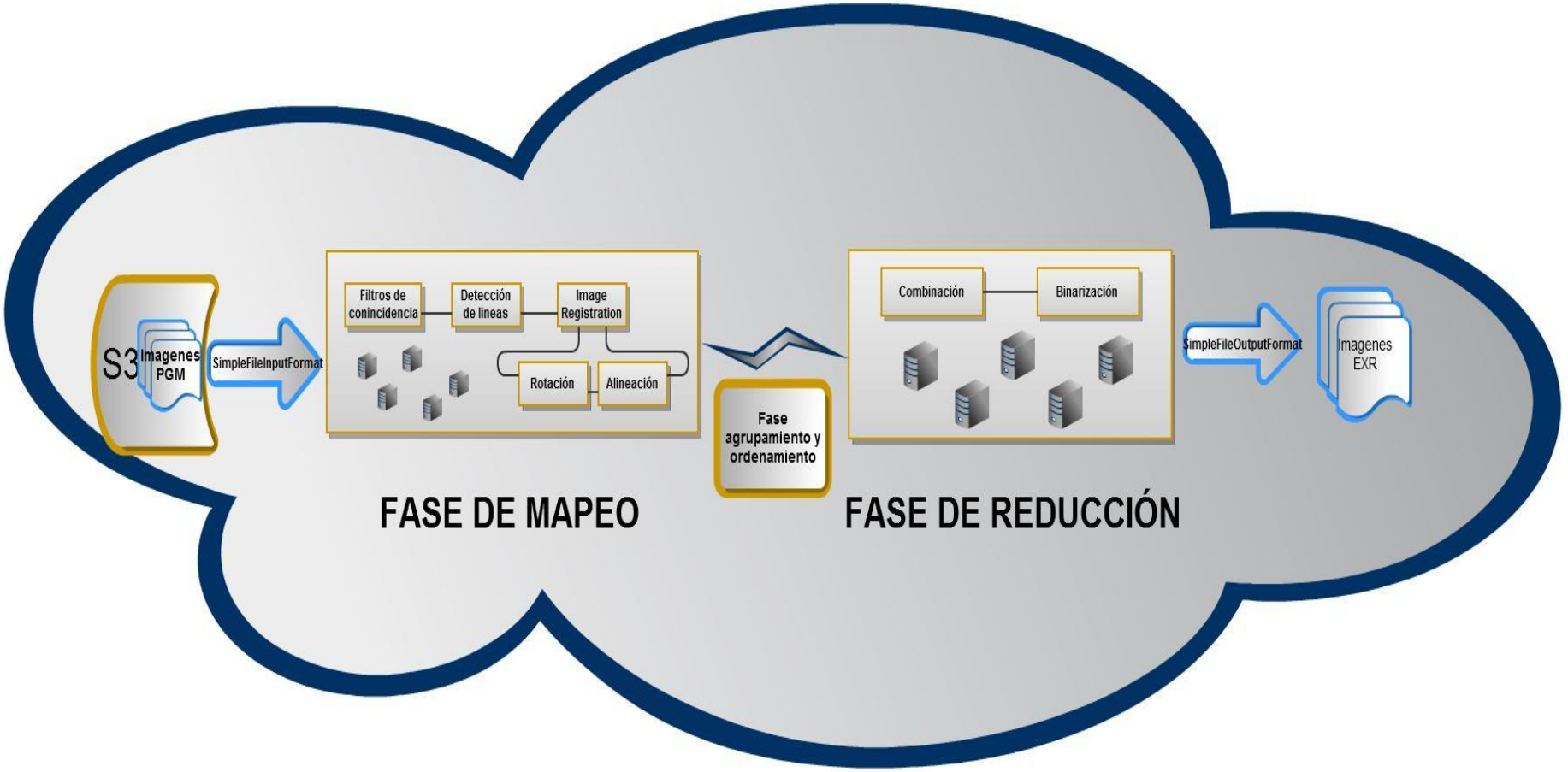
AMAZON WEB SERVICES (I)

- Servicios de computación en la nube orientado a la escalabilidad:

  - Elastic Computing Cloud (EC2)
    Permite levantar nodos bajo demanda proveyendo la infraestructura.
  - Simple Storage Service (S3)
    Almacenamiento escalable de datos
  - Elastic MapReduce (EMR)
    Automatiza tarea de levantar un clúster Hadoop sobre EC2

AMAZON WEB SERVICES (II)

DISEÑO


S3
Imagenes PGM
SimpleFileInputFormat
Filtros de conincidencia
Detección de lineas
Image Registration
Rotación
Alineación
Fase agrupamiento y ordenamiento
Combinación
Binarización
SimpleFileOutputFormat
Imagenes EXR
FASE DE MAPEO
FASE DE REDUCCIÓN

IMPLEMENTACIÓN (I)

- Implementación de las clases SimpleFileInputFormat y SimpleFileOutputFormat en java para la lectura y escritura de imágenes completas en Hadoop.

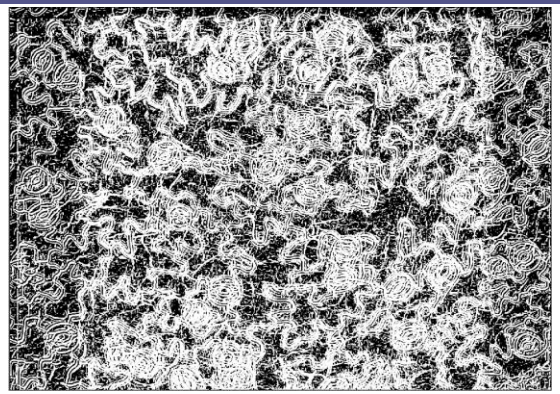
- Implementación de la librería ImageOpenCV con funciones de procesamiento de imágenes que trabaja en conjunto con la librería Stira de Filip Rooms.

IMPLEMENTACIÓN (II)

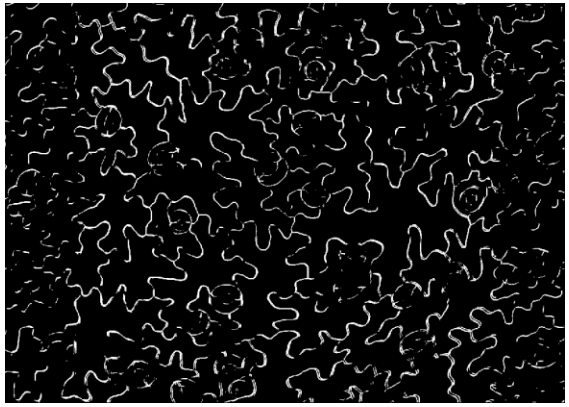
- Modificación del algoritmo de detección de líneas propuesto por Steger de la librería Stira.

- Implementación de funciones para leer y escribir imágenes de punto flotante con la librería OpenEXR.

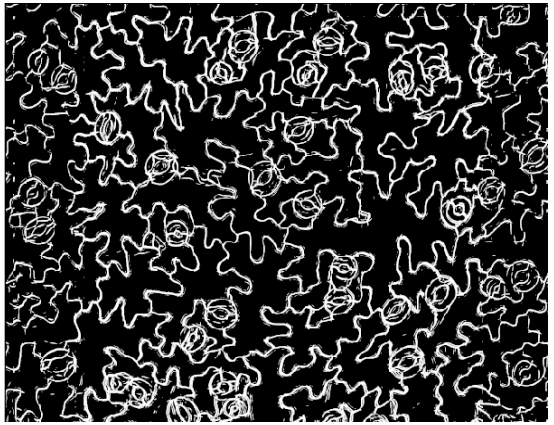
PRUEBAS Y RESULTADOS (I)



Resultado de usar un umbral de 0.0007.



Resultado de usar un umbral de 0.02.

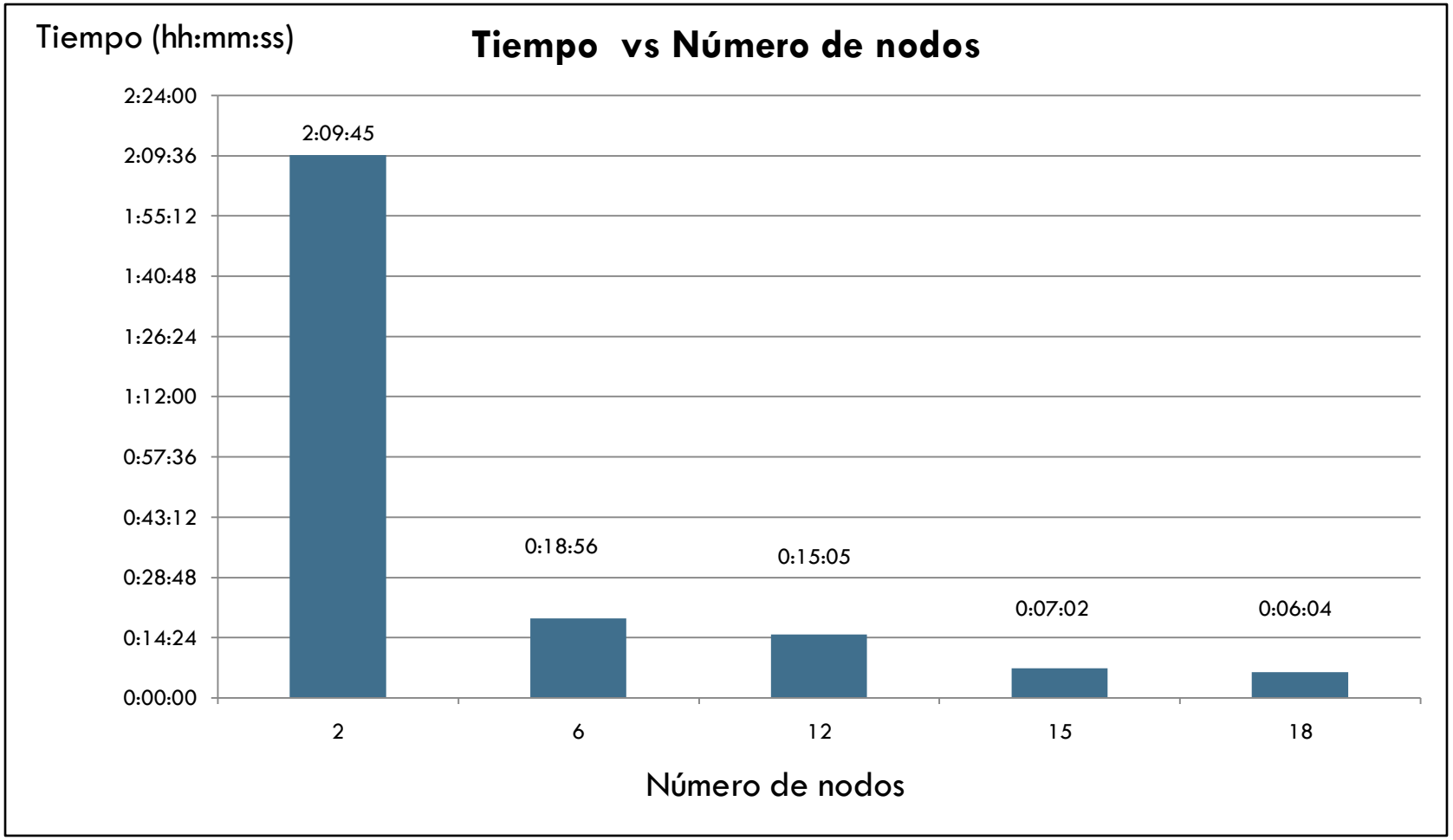


Resultado de usar un umbral de 0.005.

PRUEBAS Y RESULTADOS (II)

	Matlab			C++	
Steps	Images	Time x image	Total time	Time x image	Total time
Image fusion	24	5	120	0	0
Matching	12	51	612	9	108
Line detection (2 scales)	12	13	156	17	208
Registration (300x300 patch)	11	5	55	2	22
Merging	12	1.7	20.4	1	12
Symmetry (1 radii value)	1	27	27	1	1
			990.4 seg		**351** seg
			16.5066667 min		**5.85** min

PRUEBAS Y RESULTADOS (III)



PRUEBAS Y RESULTADOS (IV)

Nodos	Costo($)	Duración
2	1.47	2:09:45
6	1.47	0:18:56
12	2.67	0:15:05
15	3.27	0:07:02
18	3.87	0:06:04

COSTO DE GRUPO = Σ COSTOS de TODAS LAS SESIONES

COSTO DE UNA SESIÓN = $0.20 * # nodos en cluster * # de horas en que el cluster estuvo levantado (se redondean al entero superior) + $ 0.10 por GB transferido hacia el cluster + $ 0.17 por GB transferido desde el cluster.

CONCLUSIONES

• El valor de umbral que brinda información efectiva de las paredes de las células se encuentra alrededor de 0.005.

• El algoritmo implementado en C++ es aproximadamente 3 veces más rápido que el algoritmo desarrollado en Matlab.

• El número ideal de nodos para con un volumen de datos a procesar mayor a 700 MB, es de 6 nodos.

RECOMENDACIONES

• Una de las posibles mejoras del proyecto es crear una interfaz para la carga de las imágenes, ejecución del algoritmo y descargar de las imágenes de salida.

• Es recomendable utilizar un AMI que tenga instalada la misma distribución con la que se realizaron las pruebas locales.

• Configurar Hadoop para que la salida de la aplicación sea guarda directamente en un bucket de S3 para una mejor accesibilidad a la información.

¿PREGUNTAS?

Gracias por su atención