IMPLEMENTACIÓN EN PARALELO DEL ALGORITMO DE SEGMENTACIÓN DE IMÁGENES SPLIT & MERGE

ANDREA DEL PILAR REYES MENDOZA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2011

IMPLEMENTACIÓN EN PARALELO DEL ALGORITMO DE SEGMENTACIÓN DE IMÁGENES SPLIT & MERGE

ANDREA DEL PILAR REYES MENDOZA

Tesis de Grado para optar por el título de
Ingeniera de Sistemas

Director
Msc. Víctor Eduardo Martínez Abaunza

Codirectora
Msc. Ana Beatriz Ramírez Silva

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2011

Se nos han dormido los amaneceres
se nos han escondido los atardeceres
se nos han olvidado las noches estrelladas...

En un cielo blanco de Luna llena
aparecen esos viejos recuerdos
como sacados de un sótano
qué añejos están...
y una triste lágrima se resbala contra el viento
hacia el terreno árido
en el que jamás volverá a renacer una flor...

Podríamos tratar de revivirlos
buscando en un mar oscuro de lamentos
pero jamás volveremos a encontrar
aquella luz que algún día navegó cautiva
y seductora por nuestros ojos
y que por descuidados la dejamos naufragar...

☀ Niña de la Luna ☾

A mis padres Edilma y Rafael, a mis hermanitas Nubia y Yenny, a mi hermanito Yeisson, los quiero mucho mucho y les agradezco todo su esfuerzo :)

A Dario por ser un gran amigo, gracias por todo :P

A la Niña de la Luna por llenar con sus escritos mi vida de poesía, sueños y amor ☪

A mis súper amigos Tania, Raquel, Mónica, Daniel, Jenny, Ariel gracias por apoyarme :D

A mi director de proyecto Victor, gracias por esperarme :P

A mis compañeros de carrera y profesores de universidad ;)

A todos los amigos y compañeros que conocí en todos los grupos, organizaciones, cursos y demás actividades extracurriculares a los que pertenecí antes de que entregara este compendio, la pasé muy bien y aprendí más de la vida que en todos los años de U :)

A todos ellos les agradezco formar parte de mi vida y mi corazón, me han ayudado a edificar lo que ahora soy y espero que sea para bien...

Andrea Reyes

Índice General

Índice de Figuras

Índice de Tablas

Índice de Gráficas

RESUMEN

Título: IMPLEMENTACIÓN EN PARALELO DEL ALGORITMO DE SEGMENTACIÓN DE IMÁGENES SPLIT & MERGE.*

Autora: Andrea del Pilar Reyes Mendoza **

Palabras Claves: Algoritmos Paralelos, Segmentación, Clusters, Algoritmo Split & Merge, Cáncer Cervical.

En el Grupo de Investigación en Ingeniería Biomédica (GIIB), se han realizado trabajos de investigación con estudiantes de pregrado y posgrado en el campo del Procesamiento de Imágenes Digitales en la detección del Cáncer Cervical, sin embargo, se han presentado inconvenientes en la etapa de segmentación, ya que las imágenes a procesar son de gran tamaño y calidad, por lo que el tiempo empleado por los algoritmos utilizados para la segmentación de imágenes es muy elevado, y se tiene la dificultad para determinar la ubicación de la frontera final debido a que los recursos computacionales son insuficientes para el procesamiento requerido.

Como alternativa de solución se propuso el uso de la Computación de Alto Rendimiento por medio de la arquitectura de computador paralelo Cluster, por ser una tecnología de bajo precio, acorde a los recursos económicos disponibles, y de esta forma poder disminuir la carga computacional y el tiempo de procesamiento para segmentar una imagen Cervico-Uterina.

En la presente investigación se ha desarrollado un algoritmo paralelo para segmentar imágenes, basado en el método de regiones Split & Merge. El diseño del algoritmo se llevó a cabo siguiendo la metodología de Ian Foster. Se implementaron tres prototipos del algoritmo con el fin de analizar la mejor forma de etiquetar las regiones, la eficiencia en el esquema de comunicaciones y el balanceo de carga al momento de agrupar y asignar tareas a los procesadores. Para determinar la homogeneidad de la región al momento de evaluar el criterio de homogeneidad se realizó mediante pruebas estadísticas. Por último se analizaron los resultados de las pruebas en cuanto al rendimiento y la eficiencia, obteniendo datos satisfactorios.

---

* Trabajo de Investigación

** Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Víctor Eduardo Martínez Abaunza. Codirectora: Ana Beatriz Ramírez Silva.

ABSTRACT

Title: PARALLEL IMPLEMENTATION THE ALGORITHM DIGITAL IMAGES SEGMENTATION SPLIT & MERGE.*

Author: Andrea del Pilar Reyes Mendoza **

Keywords: Parallel Algorithms, Segmentation, Clusters, Split & Merge Algorithm, Cervical Cancer.

In the Biomedical Engineering Research Group (GIIB), has been performed research works in undergraduates and postgraduates in the field of Digital Image Processing in the detection of Cervical Cancer, however there have been problems in the stage of segmentation, the images to be processed are of great size and quality, so the time spent by the algorithms used for image segmentation is very high, and there is the difficulty in determining the location of the final frontier because computational resources are insufficient for the required processing.

As an alternative solution is proposed using the High Performance Computing through Cluster parallel computer architecture, being a low-cost technology according to economic resources available, and thus may reduce the computational load and processing time for segmenting an image cervical cancer.

In this research we have developed a parallel algorithm to segment images based on the method of regions Split & Merge. The design of the algorithm is carried out following the methodology of Ian Foster. Three prototypes were implemented algorithm to analyze the best way to label the regions, the efficient communication scheme and load balancing when to group and assign tasks to processors. To determine the homogeneity of the region when evaluating the criterion of homogeneity was performed using statistical tests. Finally, we analyzed the test results in performance and efficiency, obtaining satisfactory data.

---

* Researching Work

** Physical Mechanics Engineering Faculty. Informatics and Systems Engineering School. Director: Víctor Eduardo Martínez Abaunza. Co-director: Ana Beatriz Ramírez Silva.

Capítulo 1

Introducción

El Cáncer Cervical es la principal causa de muerte por neoplasia en mujeres de países en vía de desarrollo y la segunda a nivel mundial [1]. La lenta evolución de la enfermedad y la accesibilidad de células del cérvix para su estudio, permite tener tiempo y herramientas para detectar y erradicar la enfermedad, si el diagnóstico se hace oportunamente, lo que hace que el Cáncer sea una neoplasia 100% prevenible. [2]

La citología es la prueba más utilizada para la detección oportuna de la enfermedad y con apoyo del procesamiento de imágenes digitales se han implementado sistemas software con el fin de reducir el número de falsos negativos. Apoyarse en la tecnología es una opción que tienen los médicos para hacer un buen diagnóstico, lo que ha aumentado el auge en la utilización y por ende la investigación y desarrollo de software médico, por esta razón es importante avanzar con este tipo de herramientas, en este caso para prevenir el cáncer cervical.

Herramientas software utilizadas para la caracterización de células endocervicales en imágenes digitales tomadas de una citología cérvico uterina, han presentado

[1] Eluf-Neto J, Nascimento CM. Cervical Cancer in Latin America. Sem Oncol 2001;28:188-97

[2] Kiviat N. Natural history of cervical neoplasia: overview and update. Am J Obstet Gynecol 1996; 175:1099-104.

inconvenientes en la ejecución de los algoritmos para segmentar las imágenes, el tiempo de procesamiento es considerablemente alto. [3]

Trabajos de investigación sobre el tema se han realizado en el Grupo de Investigación en Ingeniería Biomédica (GIIB)[4], en los cuales se ha comprobado el tiempo elevado en correr un algoritmo de segmentación de imágenes[5]. Para resolver este problema, se propuso el uso de la Computación de Alto Rendimiento en investigaciones anteriores. [6]

Dando continuidad a la línea de investigación de Procesamiento de Imágenes Digitales en la detección del Cáncer Cervical del GIIB, se planteó el presente proyecto, el cual busca responder los interrogantes: ¿Es recomendable el uso de la Computación de Alto Rendimiento para resolver este tipo de problema?, ¿En cuanto se puede reducir el tiempo de análisis en la segmentación de una imagen digital de una muestra cervical, con el desarrollo de un algoritmo paralelo, respecto a la versión secuencial?.

Para resolver estas preguntas se diseñó e implementó un algoritmo que realiza la segmentación de la imagen por medio del método de regiones Split & Merge, se analizó el rendimiento, llegando a la conclusión de que los resultados obtenidos son aceptables.

Con la implementación de este algoritmo se observa que es una alternativa que se acomoda más a la realidad del país y adicionalmente se busca promover un método más económico para el procesamiento de imágenes médicas.

[3] [Mar07b] p. 93.
[4] www. Sis25.uis.edu.co
[5] [Mar07b] p. 93. [A+06b] [N+06]
[6] [Mar07b] p. 93.

1.1. Objetivos

1.2.1. Objetivo General

Implementar una versión en paralelo del algoritmo basado en el método de regiones Split & Merge para el proceso de segmentación de imágenes digitales de muestras Cervico-Uterinas.

1.2.2. Objetivos Específicos

- Aplicar la metodología del diseño de algoritmos paralelos al algoritmo de crecimiento de regiones.
- Analizar y definir los recursos hardware y las herramientas software a utilizar en el proyecto para optimizar el tiempo de procesamiento.
- Desarrollar un prototipo a partir de la implementación de una versión paralela del algoritmo de crecimiento de regiones Split & Merge.
- Verificar y evaluar el resultado de la segmentación realizada por el algoritmo paralelo en imágenes médicas (muestras cervico-uterinas).
- Validar los tiempos de ejecución entre la versión secuencial y la versión paralela.

1.2. Justificación y Planteamiento del Problema

El procesamiento digital de imágenes ha sido utilizado en una gran cantidad de disciplinas científicas, con diferente grado de éxito. Debido a la disminución en el costo de los dispositivos para adquisición de imágenes y al aumento en las prestaciones de los equipos de cómputo, dichas disciplinas han encontrado una nueva perspectiva de desarrollo tecnológico, brindando a su crecimiento una herramienta que se adapta tanto a sus necesidades como a su presupuesto, por consiguiente se puede afirmar que su crecimiento continuará vigente.

La segmentación de imágenes es una de las etapas más importantes en la visión computacional. Los algoritmos de segmentación pueden basarse en la discontinuidad o en la similitud de los píxeles. En el primer caso, las principales áreas de interés son la detección de puntos aislados y la detección de bordes y líneas; en el segundo, es usual utilizar algoritmos como la umbralización o aquellos orientados a regiones (crecimiento ó Split & Merge), dado que agrupan los píxeles de acuerdo con propiedades semejantes.

Por otra parte, en el método basado en regiones, aunque utilizan propiedades referentes a la homogeneidad de las imágenes digitales para determinar la localización de la frontera final, tiene el inconveniente de un alto consumo de recursos computacionales, lo cual es especialmente relevante en aplicaciones que requieren respuesta en tiempo real, y por esta razón, los algoritmos paralelos constituyen un enfoque muy importante para resolver este problema.

El cáncer Cervico-Uterino es la segunda causa de muerte después todos los cánceres en Colombia, y después del cáncer de seno, es el cáncer que más afecta a las mujeres. La prueba de citología Cervico-Uterina se utiliza para establecer la población de mujeres con riesgo de sufrir este tipo de cáncer, el

principal inconveniente al implantar un sistema de procesamiento de imágenes para apoyar el análisis, es que las imágenes adquiridas en la prueba son de gran tamaño y cantidad, y el tiempo en la etapa de segmentación es bastante elevado.
Ante esta situación se propone la paralelización del algoritmo mencionado, con el fin de evaluar la reducción en el tiempo de ejecución; pues en la actualidad no se ha logrado una implementación secuencial óptima, debido a que en diferentes pruebas realizadas se ha llegado a utilizar la totalidad de los recursos computacionales disponibles sin que se obtenga un resultado.

1.3. Estado del Arte

Se citan algunos trabajos realizados respecto al tema de paralelización del algoritmo de segmentación de regiones Split & Merge, los cuales se tomaron como guía para el desarrollar el presente trabajo de investigación:

- The load unbalancing problem for region growing image segmentation algorithms. [7]

Este documento analiza y evalúa las implementaciones paralelas de un algoritmo de segmentación basado en la estrategia Split-and-Merge. Se proponen y analizan diferentes estrategias para la selección de los identificadores de la región y su influencia en el tiempo de ejecución y la distribución de la carga en los procesadores.

[7] [MGG03]

- Implementation of a Region Growing Algorithm on Multicomputers: Analysis of the Work Load Balance. [8]

En los algoritmos paralelos se presenta un problema de fluctuaciones imprevisibles de carga en los procesadores. En este trabajo se propone tanto una estrategia de equilibrio de carga estático y una de carga dinámico, para combatir los desbalanceos de carga en cada iteración de un algoritmo paralelo.

- Parallel implementation of an adaptive split-and-merge method for medical image segmentation. [9]

Para la prueba de homogeneidad de la región, se lleva a cabo mediante una técnica de análisis de función localizada y pruebas estadísticas. La técnica de análisis de función combina un co-ocurrencia de la matriz y el histograma de sus elementos cerca de la diagonal para el cálculo de los valores límite de la desviación media estándar, el contraste de grises, y la razón de verosimilitud. A continuación, se determina si dos regiones deben dividirse o se fusionaron en la formación de regiones final.

[8] [MGG00]
[9] [JCSW92]

1.4. Estructura del Documento

Capítulo 2. Este capítulo está dedicado a los fundamentos teóricos necesarios para el desarrollo de la investigación.

Capítulo 3. En este capítulo se describe la metodología que se utilizó para el desarrollo del proyecto para comprender mejor la organización de los siguientes capítulos.

Capítulo 4. Este capítulo presenta el diseño del algoritmo realizado.

Capítulo 5. En este capítulo se describe la implementación del algoritmo respecto a el diseño propuesto y la metodología de desarrollo.

Capítulo 6. En este capítulo se presentan las pruebas a las cuales fueron sometidos los prototipos del algoritmo implementado y sus resultados.

Capítulo 7. Este capítulo se expone el análisis de los resultados obtenidos en las pruebas.

Capítulo 8. En este capítulo se deducen las conclusiones de la investigación y las recomendaciones para una futura continuación de la investigación en el tema.

Capítulo 2

Fundamentos Teóricos

En este capítulo se propone una revisión de bases conceptuales para entender el desarrollo de la investigación, se presentan como una ayuda al lector de forma que sea autocontenido. Primero un breve repaso de Procesamiento de Imágenes Digitales y la importancia de la etapa de Segmentación. Luego se hace énfasis en el método de crecimiento de regiones como mejor opción para segmentar y se hace una descripción detallada del algoritmo Split & Merge propuesto por Horowitz and Pavlidis, dicho algoritmo de crecimiento de regiones seleccionado a paralelizar en la investigación. Por consiguiente se expone el tema de aplicación sobre el cual se seleccionaron las imágenes médicas para analizar. Y por último se sustentan los conceptos de Algoritmos Paralelos y Computación de Alto Rendimiento.

2.1. Procesamiento de Imágenes Digitales

Una imagen puede ser definida como una función de dos dimensiones, f (x, y), donde x e y son las coordenadas espaciales, la amplitud de f en cualquier par de coordenadas (x, y) se llama la intensidad o nivel de gris de la imagen en ese punto. Cuándo x, e, y los valores de amplitud de f son todos finitos, cantidades discretas, le llamamos a la imagen una imagen digital.

Una imagen digital se compone por un número finito de elementos, cada uno de los cuales tiene una particular localización y valor. Estos elementos se denominan elementos de imagen y píxeles. Píxel es el término más utilizado para denotar los elementos de una imagen digital. [10]

La visión es el más avanzado de nuestros sentidos, por lo que las imágenes desempeñan un papel importante en la percepción humana. Sin embargo, a diferencia de los seres humanos, que se limitan a la banda visual de la espectro electromagnético (EM), las máquinas de proyección de imágenes cubren casi todo el espectro electromagnético, que van desde gamma a ondas de radio. Pueden funcionar en las imágenes generadas por las fuentes que los seres humanos no están acostumbrados a asociar con imágenes. Estos incluyen ultra-sonido, microscopio electrónico e imágenes generadas por ordenador, el procesamiento de imágenes digitales abarca un campo amplio y variado de aplicaciones.

El proceso de adquisición de una imagen, el preprocesamiento de esa imagen, la extracción (segmentar) de los caracteres individuales y el reconocimiento, están en el ámbito de lo que llamamos el Procesamiento de Imágenes Digitales.

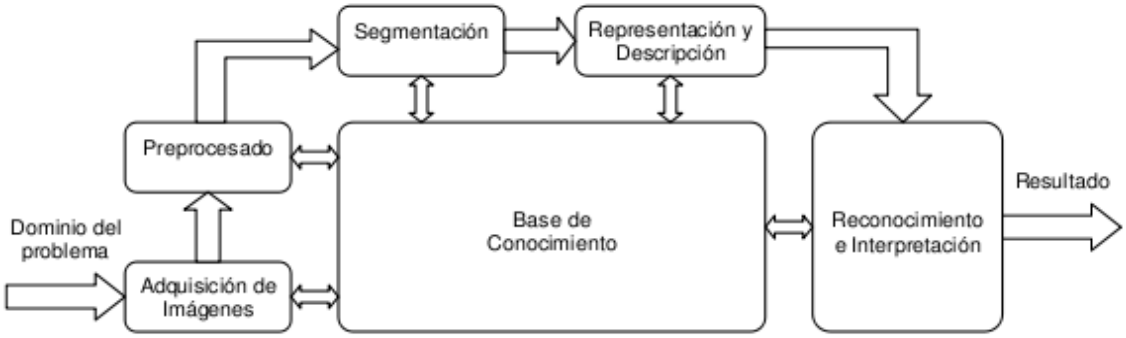
2.1.1. Etapas del Procesamiento Digital de Imágenes

La palabra digital hace referencia al cálculo por métodos numéricos, o por unidades discretas, y conforme a la definición dada anteriormente, una imagen digital es una representación numérica de un objeto donde los píxeles son las unidades discretas y la escala de gris cuantificada provee las componentes para el cálculo numérico.

[10] [GW02] p.15-39.

De acuerdo con su definición, la palabra procesamiento consiste en someter algo a un proceso, que es una serie de acciones u operaciones para obtener un resultado deseado. En ese orden de ideas, se puede definir el procesamiento digital de imágenes como la serie de operaciones a las cuales se somete la representación numérica de un objeto para obtener un resultado esperado. Estas acciones han sido estudiadas a lo largo del tiempo y han sido agrupadas en las siguientes etapas, de acuerdo con la figura 1.

Figura 1 . Etapas fundamentales del procesamiento de imágenes.



Ibid., p. 8.

- Adquisición: Comprende todas las tareas para obtener la imagen digital, por diferentes dispositivos electrónicos (sensibles a la luz o a cualquier banda del espectro electromagnético).
- Preprocesamiento: Aquí se agrupan las operaciones numéricas que tienden a mejorar la información presente en la imagen. Existen métodos espectrales basados en la información frecuencial y otros operan directamente sobre el dominio espacial de la imagen.
- Segmentación: Cada una de los objetos que constituyen la imagen deben ser separados para su respectivo análisis. Mediante técnicas estadísticas y

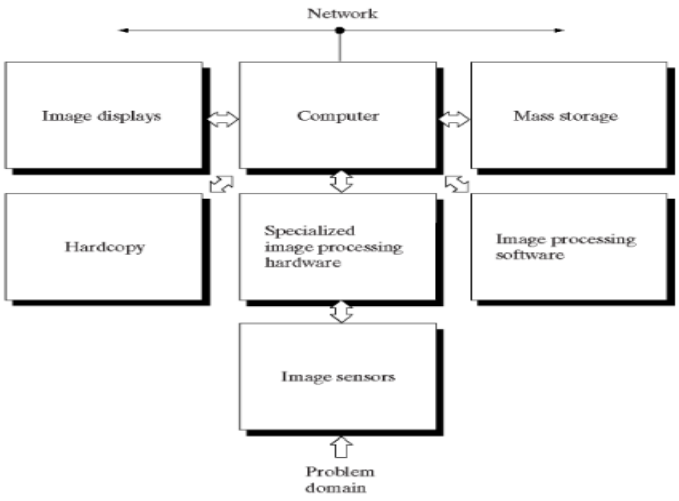
otros métodos pueden diferenciarse cada una de las regiones u objetos que componen una imagen digital.

- Descripción: Los objetos o regiones presentes en una imagen poseen características que los diferencian de los demás. Lo que se busca en esta etapa es asignarle propiedades a los objetos que contribuyan con el análisis de la imagen.
- Interpretación: Una vez que se obtienen las propiedades de los objetos se utilizan técnicas de reconocimiento de patrones o se emplean algoritmos que clasifiquen los objetos presentes en la imagen.

2.1.2. Componentes de un Sistema de Procesamiento de Imágenes

La Figura 2 muestra los componentes básicos que incluye un sistema típico de uso general utilizado para el procesamiento de una imagen digital.

Figura 2. Componentes de un sistema de procesamiento de imágenes



Gonzalez, R. Woods, R. p. 29

Se requieren dos elementos para adquirir imágenes digitales. El primero es un dispositivo físico que es sensible a la energía radiada por el objeto que queremos de la imagen. El segundo, llamado un digitalizador, es un dispositivo para convertir la salida del dispositivo físico de detección en forma digital.

El hardware especializado de procesamiento de la imagen por lo general consiste en el digitalizador mencionado, además de hardware que realiza otras operaciones primitivas, como un unidad aritmética lógica (ALU), que realiza operaciones aritméticas y lógicas en paralelo sobre imágenes.

El Software para el procesamiento de imágenes se compone de módulos especializados que realizan tareas específicas. Un paquete bien diseñado también incluye la capacidad para que el usuario escriba código, como mínimo, utiliza los módulos especializados. Paquetes más sofisticados de software permiten la integración de los módulos y comandos de software de propósito general de por lo menos un lenguaje de programación.

La capacidad de almacenamiento masivo es una necesidad en aplicaciones de procesamiento de imágenes. El almacenamiento digital para aplicaciones de procesamiento de la imagen se divide en tres categorías principales: a corto almacenamiento a largo plazo para su uso durante el proceso, en línea de almacenamiento para recordar relativamente rápido , y almacenamiento de archivos, que se caracteriza por un acceso poco frecuente.

Las imágenes usadas el día de hoy son principalmente de color (preferentemente de pantalla plana) monitores-TV. Los monitores son impulsados por los resultados de imagen y tarjetas de gráficos de la pantalla que son una parte integral del sistema informático.

Para la copia de los dispositivos de grabación de imágenes se incluyen las

impresoras láser, cámaras de filmar, los dispositivos sensibles al calor, las unidades de inyección de tinta, y las unidades digitales, como ópticas y CD-ROM de los discos.

La creación de redes es casi una función predeterminada en cualquier sistema informático en uso hoy en día. Debido a la gran cantidad de datos inherentes a las aplicaciones de procesamiento de imágenes, la consideración clave en la transmisión de la imagen es de ancho de banda.

2.2.3. Segmentación de Imágenes Digitales

La segmentación subdivide una imagen en sus regiones constituyentes u objetos, el nivel al que la subdivisión se realiza depende del problema a resolver. Es decir, la segmentación debe detenerse cuando los objetos de interés en una aplicación han sido aislados. No tiene sentido la realización de segmentación más allá del nivel de detalle necesario para identificar a los elementos.

La segmentación de las imágenes es una de las tareas más difíciles en el procesamiento de imágenes. La precisión de la segmentación determina el éxito o el fracaso de los procedimientos de análisis computarizado.

La segmentación es el proceso de asignación de píxeles a cada región que tiene propiedades comunes, con atributos de imagen como la intensidad del píxel, valores espectrales y las propiedades de textura.

Algoritmos de segmentación de la imagen en general se basan en una de las dos propiedades básicas de los valores de intensidad: la discontinuidad y similitud. En la primera categoría, el enfoque es dividir una imagen basada en los cambios

bruscos en la intensidad, tales como los bordes de una imagen. Los enfoques principales de la segunda categoría se basan en la partición de una imagen en regiones que son similares de acuerdo a un conjunto de criterios predefinidos.

Sea R representa la imagen completa que tiene diferentes objetos. Una segmentación de R es una partición de R en subconjuntos R_1, R_2,…, R_n, tal que: [11]

(a) $U_{i=1}^{n} R_i = R,$

(b) $R_i \text{ is a connected region, } \forall\ i,$

(c) $R_i \cap R_j = \emptyset\ \forall\ i,j, i \neq j,$

(d) $P\left(R_i\right) = TRUE\ \forall\ i,$

(e) $P\left(R_i \cup\ R_j\right) = FALSE \forall\ i, j, i \neq\ j,$

Donde $P\left(R_i\right)$ es un predicado lógico sobre el conjunto de píxeles en el conjunto de píxeles en R_i y $\emptyset$ es el conjunto vacío.

La proposición (a) indica que la segmentación debe ser completa, es decir, cada píxel debe estar en una región, mientras que la segunda indica los píxeles pertenecientes a una región debe estar conectado. La proposición (c) representa que las regiones deben ser desarticulada, y (d) se ocupa de las propiedades que deben ser satisfechas por los píxeles en todas las regiones segmentadas R_i de tal forma $P(R_i)$ = TRUE si todos los píxeles de R_i son equivalentes con respecto R_i y R_j son diferentes en el sentido de predicado P.

[11] [GW02] p.612

2.2.4. Método de Regiones Split & Merge

El algoritmo de Split & Merge, como su nombre indica, engloba dos procesos característicos, uno de división y el otro de fusión.

SPLIT. A partir de la imagen original, que es considerada como el primer gran bloque, se procede a aplicar el split, es decir, se le aplica un determinado criterio de homogeneidad por el que, si dicho bloque no lo cumple, se divide geométricamente en cuatro partes, que a continuación serán analizados por el mismo criterio. Dicho proceso se repite hasta que todos los bloques cumplan el criterio de división.

MERGE. Una vez finalizada la fase de split, se procede ha realizar la de Merge, fase que une los bloques creados. En este caso, se ha de comprobar si dos bloques consecutivos adyacentes, es decir, que estén en contacto, cumplen el criterio anterior. Si es así, dichos bloques quedarán fusionados, formando un único bloque. Dicho proceso se repite hasta llegar a la idempotencia, es decir, el momento en el cual ya no pueden fusionarse más bloques.

Las principales ventajas de este algoritmo son, en primer lugar, su sencillez y simplicidad, tanto el algoritmo en sí ,como la partición inicial desde la que se inicia el mismo (imagen original como primer gran bloque a tratar). Por otro lado, se tiene una visión global, y no local, de como va evolucionando el algoritmo.
El principal inconveniente, es sin duda,la forma que tiene de segmentar, de dividir los bloques, pues es totalmente geométrica, obteniéndose contornos no muy naturales.

El Algoritmo de Split & Merge propuesto por Horowitz y Pavlidis [12] es uno de los algoritmos más populares para la segmentación de imágenes.

El algoritmo puede resumirse en los siguientes pasos:

(1) Dividir cualquier región R_i en cuatro cuadrantes disjuntos donde $P(R_i)$ = FALSE.
(2) Fusionar las regiones adyacentes R_i y R_j para las que $P(R_i \cup R_j)$ = TRUE.
(3) Detener cuando no sea posible realizar más fusiones o divisiones.
De lo contrario, repita los pasos (1) y (2).

La etapa de división consiste en generar un quad-tree (ver Fig. 3.), Donde los nodos en el nivel i son el resultado de la división de las sub-imágenes de tamaño $N_1 / 2^i \times N_2 / 2^i$ en cuatro sub-imágenes de tamaño $N_1 / 2^{(i+1)} \times N_2 / 2^{(i+1)}$. La construcción de este quad-tree puede llevarse a cabo siguiendo una estrategia de Bottom-up de fusión de sub-imágenes o la estrategia de Top-Down de división.

En ambas estrategias, sólo los nodos que satisfacen el criterio de homogeneidad serán generados. La fase de Split termina cuando ya no hayan más regiones que se puedan generar. El conjunto de las regiones en el quad-tree producidos por la fase de Split será el conjunto de las regiones de entrada para la fase de mezcla.

[12] [HP74]

Algoritmo de Segmentación Split & Merge:

Entrada: Imagen I, Predicado P

Salida: Imagen segmentada IS

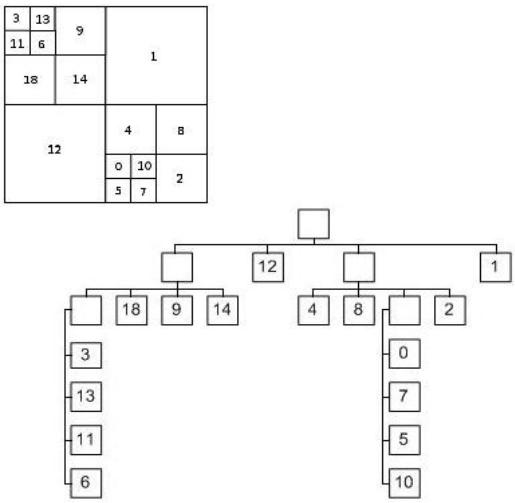
1: **Procedimiento** SegmentaciónSplitMerge(I,P)
2: Iniciar el árbol con I como nodo raíz

3: **Para cada** Región Ri iterativamente **Hacer**
4: **Si** P(Ri) = FALSO **Entonces**
5: Descomponer Ri en 4 cuadrantes
6: Actualizar el árbol cuaternario
7: **Fin Si**
8: **Fin Para**

9: **Para cada** Regiones Ri,Rj adyacentes Hacer
10: **Si** P(Ri U Rj) = V ERDADERO **Entonces**
11: Fusionar Ri y Rj
12: Actualizar el árbol cuaternario
13: Repetir hasta que no hayan mas regiones que fusionar
14: **Fin Si**
15: **Fin Para**

16: Reconstruir IS desde el árbol cuaternario.
17: **Retornar IS** , La imagen segmentada
18: **Fin Procedimiento**

Figura 3. Representación Quad-tree



Autora Documento

Complejidad del Algoritmo[13]

Para hallar la complejidad de este algoritmo podemos identificar el tamaño de los datos como el número de filas que contiene la imagen, al fin y al cabo la otra dimensión es igual, ya que para que el algoritmo funcione las imágenes deben ser cuadradas.

En el peor de los casos, la imagen se subdividirá hasta que cada región quede representada por un solo píxel. Antes de que una imagen se divida, se evalúa el criterio de homogeneidad; el cual, generalmente, toma un tiempo lineal O(n).

[13] [AS]

Esto unido con el comportamiento de divisiones binarias que se realiza sobre las regiones hace que el algoritmo de descomposición sea O(n log n).

Para la fase de fusión tomaremos en cuenta el tamaño de los datos como el número de regiones que se pretenden fusionar. Podemos darnos cuenta que en el peor de los casos, cada región sería un solo píxel, por lo que el tamaño de los datos de las dos fases es la misma, aunque no representen lo mismo.

Una manera óptima de llevar control de la adyacencia de las regiones, es construir un grafo de adyacencia representado por una matriz de adyacencia.

Manipular la matriz para detectar que regiones pueden fusionarse está en el orden de O(n2), por lo que finalmente el algoritmo de segmentación split-merge está en el orden de O(n2).

2.2. Imágenes Médicas

El campo de las imágenes médicas ha avanzado significativamente en los últimos años, convirtiéndose en una herramienta primordial en la práctica de la medicina y generando, a su vez, nuevas líneas de investigación científica que abarcan múltiples disciplinas. Esta evolución ha sido posible gracias a la difusión de áreas tradicionalmente separadas pero muy relacionadas: desde la visión por computadora y el procesamiento de imágenes, pasando por técnicas gráficas y visualización, hasta los entornos inmersivos e interactivos y dispositivos para la manipulación interactiva de la información.[14]

[14] http://verona.fi-p.unam.mx/~emoya/

Podemos destacar la proyección de imagen médica para una consideración especial debido a que la vida de las personas a menudo dependen de correcta adquisición, procesamiento e interpretación de imágenes médicas. Es importante que las personas responsables de la adquisición y procesamiento de imágenes médicas entiendan tanto la naturaleza de la materia prima con las que trabajan, y la forma en que las imágenes que producen se utilizan.[15]

El propósito de la proyección de imagen médica es dar a conocer y registrar el estado estructural o funcional del cuerpo. Sobre todo queremos ver lo que está pasando dentro del cuerpo para comprobar que todo está bien, o para averiguar por qué no todo está bien. A veces queremos un registro del estado actual del cuerpo que se hace referencia en algún momento futuro para supervisar el progreso, o falta de progreso, de una enfermedad o un tratamiento.

La mayoría de las imágenes médicas tienen la intención de revelar aspectos del cuerpo que no pueden ser observadas por inspección visual o la exploración física del exterior del cuerpo.

2.2.1. Cáncer Cervical

El cáncer cervical, un tipo de cáncer común en las mujeres, es una enfermedad en la cual las células cancerígenas se desarrollan en los tejidos del cuello uterino.
El cáncer en el cuello uterino por lo general se desarrolla lentamente. Antes de que las células cancerígenas aparezcan en el cuello uterino, los tejidos del mismo experimentan cambios en los cuales las células que no son normales comienzan a formar (lo que se conoce como displasia). Una prueba de Frotis de Papanicolau normalmente detecta estas células. Luego, las células cancerígenas comienzan a

[15] [Bou010] p.31.

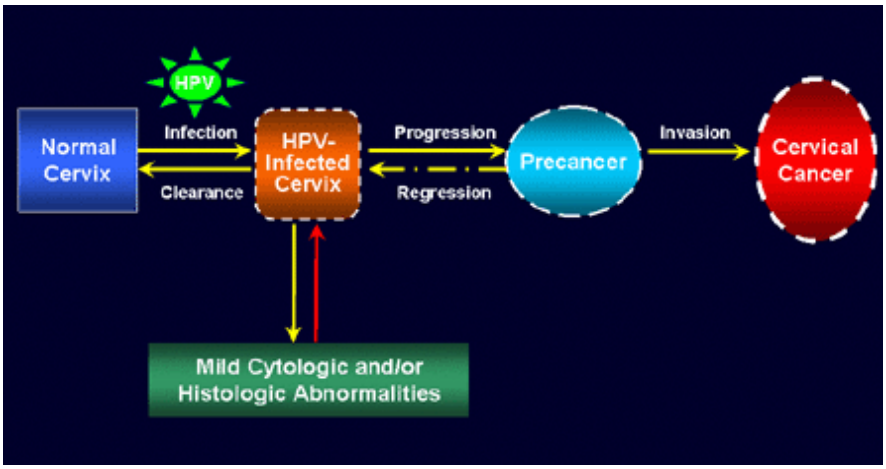
crecer y a esparcirse con mayor profundidad en el cuello uterino y en los órganos que lo rodean.

El cáncer cervical, es el segundo cáncer más común entre las mujeres en todo el mundo, es un importante problema de salud pública. había más de 493.000 nuevos casos diagnosticados y 273.500 muertes por cáncer de cuello uterino en el año 2000. Aproximadamente el 85% de estas muertes ocurrieron en países en desarrollo, y en algunas partes del mundo el cáncer de cuello uterino se cobra la vida de más mujeres que las causas relacionadas con el embarazo. Esta condición no sólo afecta a la salud y la vida de las mujeres, sino también a sus hijos, las familias y su comunidad. este impacto ampliada es a menudo infravalorado la hora de establecer las prioridades de salud y requiere una mayor consideración por las autoridades. [16]

La comprensión de cómo el cáncer cervical se desarrolla es esencial para diseñar intervenciones eficaces para prevenir las muertes por esta enfermedad. Más del 99% de los casos de cáncer cervical y sus precursores están relacionados con la infección por el VPH, una infección de transmisión sexual (ITS), que es en su mayoría asintomáticos.[17] El VPH es la ITS más común en todo el mundo, afectando a un estimado de 50 a 80% de las mujeres sexualmente activas por lo menos una vez en su vida.[18] Las mujeres son en su mayoría infectados con el VPH en la adolescencia, 20 años, o 30 años. El cáncer cervical es esencialmente una complicación rara de una ITS común.

[16] [ACCP04] p. vii
[17] Walboomers et al. 1999
[18] Koutsky 1997, Crum et al. 2003

Figura 4. Etapas Cáncer Cervical



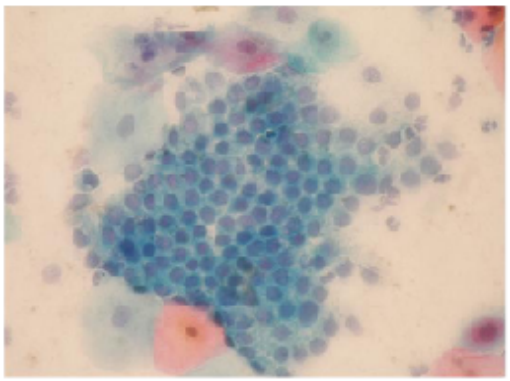
Schiffman M, Kjaer SK. J Nat Cancer Inst Manag. 2003

Los actuales programas de prevención del cáncer cervico-uterino son casi todos basados en la citología. Citología cervical es la prueba de detección que ha sido más utilizado desde mediados del siglo XX en los países desarrollados y en los países en desarrollo los que la clasificación está disponible.

La citología cervical también se conoce como la prueba de Papanicolaou, detecta células anormales en una muestra tomada del cuello del útero. Se trata de realizar un examen con espéculo para exponer el cuello uterino y recolectar células del cuello uterino con una espátula. Estas células se untan y fijan en un portaobjetos de vidrio. las diapositivas son transportadas a un laboratorio donde se procesan manualmente. Cada diapositiva es evaluado bajo el microscopio por un técnico capacitado en citología. este proceso de varias etapas puede durar varias semanas antes de que los resultados están disponibles para el cliente, aunque en programas bien organizados los resultados pueden estar disponibles pronto.

Existen dos tipos principales del cáncer de cuello uterino. Entre ocho y nueve de cada diez casos se trata de *carcinomas de células escamosas*, los cuales al observarse con un microscopio, se ve que están formados por células parecidas a las células escamosas que cubren la superficie del cuello del útero. Para el resto de los casos, la mayoría son *adenocarcinomas* que comienzan en las células glandulares que producen mucosidad. Con menor frecuencia, el cáncer tiene características de ambos tipos y se conoce como carcinoma mixto. También hay otros tipos de cáncer que se desarrollan en el cuello del útero, tal como melanoma, sarcoma y linfoma, que ocurren con más frecuencia en otras partes del cuerpo. [19]

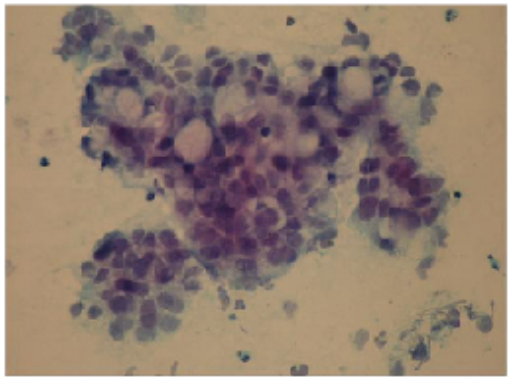
Figura 5. Células endocervicales



Martinez, V. [Mar04]

[19] American Cancer Society, http://www.cancer.org/Espanol/cancer/Cancerdecuellouterino/Resumen/resumen-sobre-el-cancer-de-cuello-uterino-what-is-cervical-cancer

Figura 6. Adenocarcinoma



Martinez, V. [Mar04]

2.3. Algoritmos Paralelos

Un Algoritmo Paralelo es un algoritmo que puede ser ejecutado por partes en el mismo instante de tiempo por varias unidades de procesamiento, para finalmente unir todas las partes y obtener el resultado correcto. Algunos algoritmos son fácilmente divisibles en partes; Por el contrario, a veces los problemas no son tan fácilmente paralelizables, de ahí que estos problemas se conozcan como problemas inherentemente secuenciales.

Los algoritmos paralelos son importantes porque es más rápido tratar grandes tareas de computación mediante la paralelización que mediante técnicas secuenciales. Esta es la forma en que se trabaja en el desarrollo de los procesadores modernos, ya que es más difícil incrementar la capacidad de procesamiento con un único procesador que aumentar su capacidad de cómputo mediante la inclusión de unidades en paralelo, logrando así la ejecución de varios flujos de instrucciones dentro del procesador.

2.3.1. Diseño de Algoritmos Paralelos de Ian Foster

La mayoría de problemas de programación tienen soluciones paralelas. La mejor solución secuencial puede ser a menudo al peor solución paralela. La metodología de diseño que se describe tiene la intención de fomentar un enfoque exploratorio de diseño en el que independiente de las cuestiones máquina. Este modelo conceptual del proceso de diseño en cuatro etapas diferentes[20]: partición, comunicación, aglomeración y asignación, en las dos primeras etapas, se centran en la concurrencia y escalabilidad y tratar de descubrir los algoritmos con estas cualidades. En la tercera y cuarta etapas, la atención se desplaza a la localidad y otros temas relacionados con rendimiento. Las cuatro etapas se ilustran en la Figura 7.

- *Particionado.* El cálculo que se va a realizar y los datos de los operados por este cálculo se descomponen en pequeñas tareas. Las cuestiones prácticas tales como el número de procesadores en el equipo de destino no se tienen en cuenta, y la atención se centra en el reconocimiento de las oportunidades para la ejecución en paralelo.

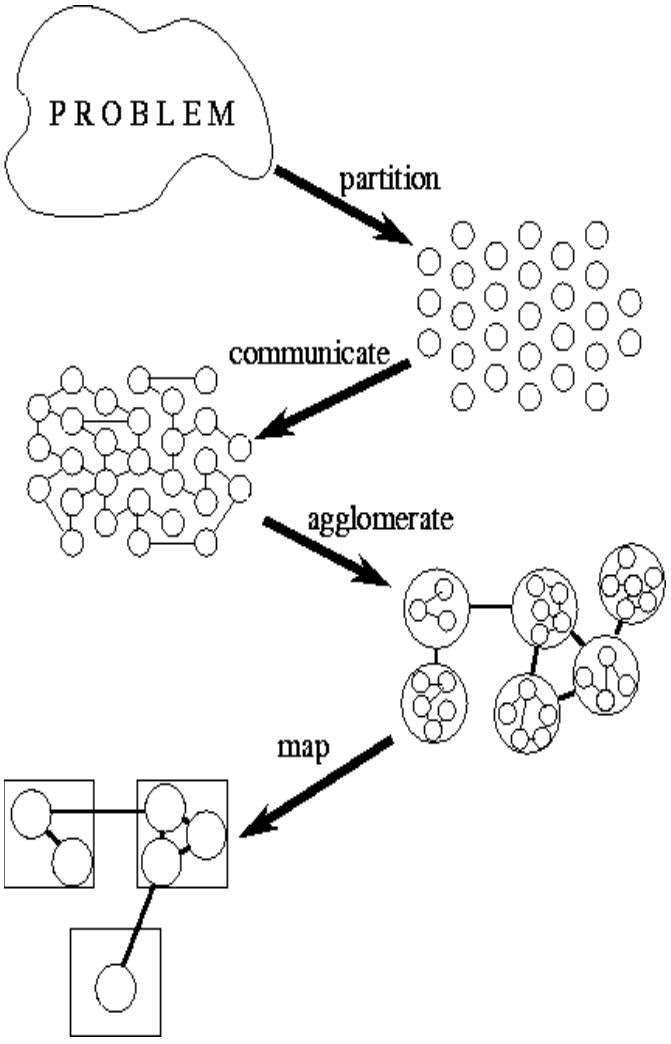
- *Comunicación.* Se determina la comunicación necesaria para coordinar la ejecución de tareas, y se escogen las estructuras comunicación apropiadas.

- *Aglomeración.* La comunicación y las tareas definidas en las dos primeras etapas de un diseño son evaluados con respecto a los requisitos de rendimiento y costes de ejecución. Si es necesario, las tareas se combinan para mejorar el rendimiento o para reducir los costes de comunicación.

[20] [Fos95]

- *Asignación.* Cada tarea se asigna a un procesador de manera que se satisfagan los objetivos de maximizar la utilización del procesador y reducir al mínimo los costos de comunicación. La asignación se puede especificar de forma estática o en tiempo de ejecución, determinado por el equilibrio de los algoritmos de carga.

Figura 7. Etapas Diseño de Algoritmos Paralelos de Ian Foster



Foster, I. 1995.

El resultado de este proceso de diseño puede ser un programa que crea y destruye las tareas de forma dinámica, utilizando técnicas de balanceo de carga para controlar la asignación de tareas a los procesadores.

2.3.1.1. Particionamiento

La etapa de partición de un diseño está pensado para exponer las oportunidades de ejecución en paralelo. Por lo tanto, la atención se centra en la definición de un gran número de pequeñas tareas a fin de producir lo que se denomina descomposición *de grano fino* de un problema.

En etapas posteriores en el diseño, la evaluación de las necesidades de comunicación, la arquitectura de destino, o problemas de software puede llevarnos a renunciar a las oportunidades para la ejecución en paralelo identificados en esta etapa.

Una buena partición divide en trozos pequeños tanto en el *cálculo* asociado con un problema y los *datos* en los que opera este cálculo. En el diseño de una partición, por lo general los programadores se centran primero en los datos asociados a un problema, a continuación, determinar una partición adecuado para los datos y, finalmente, encontrar la manera de computación asociados con los datos.

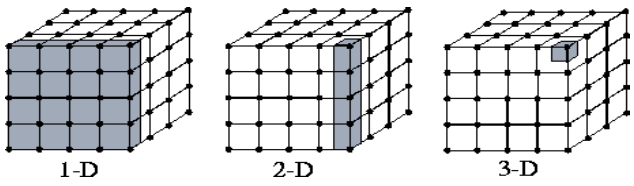
Esta partición se denomina técnica de *descomposición de dominios*. La alternativa enfoque primera descomposición de la computación a realizar y luego se ocupan de los datos se denomina *descomposición funcional*. Estas son técnicas complementarias que pueden aplicarse a los distintos componentes de un solo problema o incluso aplicado al mismo problema obtener algoritmos paralelos alternativos.

En esta primera etapa de un diseño, se trata de evitar la replicación de datos y cómputo. Al igual que granularidad, este es un aspecto del diseño que se puede volver más tarde. Puede ser que vale la pena replicar la computación o los datos, si con ello nos permite reducir las necesidades de comunicación.

Descomposición de Dominio

En este enfoque se busca primero descomponer los datos asociados a un problema. Si es posible, que se dividen estos datos en pedazos pequeños de aproximadamente el mismo tamaño. A continuación, la partición del cálculo que se va a realizar, por lo general mediante la asociación de cada operación con los datos en los que opera. Esto produce particiones en la serie de tareas, cada uno compuesto algunos datos y un conjunto de operaciones en esos datos. Una operación puede requerir datos de varias tareas. En este caso, la comunicación es necesaria para mover los datos entre las tareas. Este requisito se aborda en la siguiente fase del proceso de diseño.

Figura 8. Particionamiento Descomposición de Dominio



Foster, I. 1995.

La Figura ilustra la descomposición de un problema que tiene una grilla de tres dimensiones. Las instrucciones son ejecutadas en cada punto de la grilla repetidamente. La descomposición en el eje x, y o z se puede realizar de la forma mas fácil. La descomposición más agresiva que en este caso sería, en cada punto de la grilla. Cada tarea mantendría en un estado varios valores asociados al punto de la grilla y es responsable por el cómputo que sea necesario para actualizar su estado.

Descomposición funcional

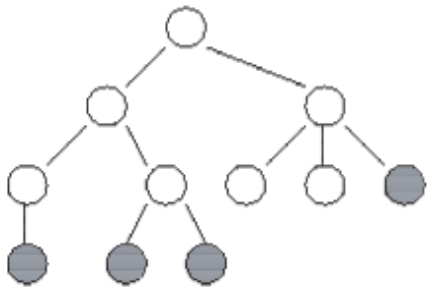
Descomposición funcional presenta una forma diferente y complementaria de pensar acerca de los problemas. En este enfoque la atención se centra en el cálculo que se va a realizar y no en los datos manipulados por el cálculo. Después de realizar la división en diferentes tareas, se procede a revisar los datos que se necesitarán para realizarla. Los requerimientos de datos pueden ser diferentes para cada tarea, en ese caso la partición ha sido completa. De lo contrario aparecen solapamientos de datos, en este caso será necesaria una cantidad de comunicación considerable. Esta es una razón por la cual se prefiere realizar una descomposición de datos.

Aunque la descomposición de dominio constituye la base para la mayoría de algoritmos paralelos, la descomposición funcional es valiosa como una forma diferente de pensar acerca de los problemas. Por esta razón, debe tenerse en cuenta al explorar posibles algoritmos paralelos. Un enfoque en los cálculos que se van a realizar a veces puede revelar la estructura de un problema, y por lo tanto las oportunidades de optimización que no sería obvia de un estudio de los datos por sí solos.

Un ejemplo de cuando la descomposición funcional puede ser la más apropiada:

una exploración en un árbol para la búsqueda del nodo “soluciones”. La descomposición del dominio seria obvia. Sin embargo una descomposición funcional de grano fino puede plantearse: inicialmente una tarea simple se crea en el nodo raíz del árbol, la tarea evalúa ese nodo, y si no es el nodo buscado crearía una nueva tarea (subárbol). Una nueva tarea sería creada cada vez que se expanda la búsqueda en el árbol (ver Figura 9)

Figura 9. Estructura de tarea para un ejemplo de búsqueda.



Foster, I. 1995.

Cada circulo representa un modulo en el árbol de búsqueda y una llamada a la tarea nueva. Las líneas representan los canales donde retorna la solución.

La descomposición funcional también tiene un papel importante que desempeñar como un programa de estructuración técnica. Una descomposición funcional que las particiones no sólo los cálculos que se va a realizar, sino también el código que realiza de que la computación es probable que reduzca la complejidad del diseño

general. Esto es a menudo el caso en los modelos computacionales de sistemas complejos, que pueden ser estructurados como colecciones de modelos más sencillos conectados a través de interfaces.

2.3.1.2. Comunicación

Las tareas que genera una partición se pretenden ejecutar concurrentemente, pero no pueden ejecutarse de forma independiente. El cálculo que se realiza de una tarea requieren los datos asociados a otra tarea. Los datos deben ser transferidos entre las tareas a fin de proceder con el cálculo . Este flujo de información se especifica en la *comunicación* de una fase de diseño.

Las comunicaciones asociadas a un algoritmo se puede especificar en dos fases. En primer lugar, definir una estructura de canal que une, ya sea directa o indirectamente, tareas que requieren datos (consumidores) con las tareas que poseen los datos (los productores). En segundo lugar, se especifican los mensajes que se envían y se reciben en estos canales. En un lenguaje paralelo de datos, basta con especificar las operaciones de datos en paralelo y la distribución de datos. Sin embargo, pensar en términos de tareas y los canales nos ayuda a pensar cuantitativamente acerca de los problemas asignación y los costos de comunicación.

La definición de un canal implica un costo intelectual y el envío de un mensaje supone un coste físico. Por lo tanto, se debe evitar la introducción de canales innecesarios y acciones de comunicación. Además, se busca optimizar el rendimiento mediante la distribución de las operaciones de comunicación a través de muchas tareas y la organización de acciones de comunicación de una manera que permita la ejecución simultánea.

Las comunicaciones se pueden clasificar como:

- *Local/Global.* Comunicación local cada tarea se comunica con un pequeño grupo de tareas “vecinas”, en contraste la comunicación global requiere que cada tarea se comunique con un gran número de tareas.
- *Estructurada/No estructurada.* Comunicación estructurada la tarea y sus vecinos forman una estructura regular como un árbol o una grilla; mientras que en la comunicación irregular la comunicación pueden ser grafos arbitrarios.
- *Síncrono/Asíncrono.* En la comunicación sincrónica productores y consumidores ejecutan la tarea en forma coordinada, mientras que en la comunicación asíncrona el consumidor obtiene los datos sin ayuda del productor.

Comunicación Local

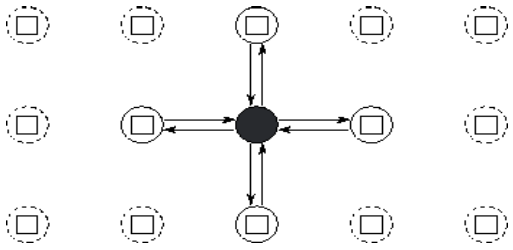
Una estructura de comunicación local se obtiene cuando una operación requiere datos de un pequeño número de otras tareas. Es entonces fácil de definir los canales que unen el trabajo responsable de realizar la operación con las tareas de explotación de los datos necesarios y la introducción adecuada enviar y recibir operaciones en las tareas de producción y de consumo, respectivamente.
A título ilustrativo, consideramos las necesidades de comunicación asociados con un simple cálculo numérico, es decir, un método de diferencias finitas Jacobi. En esta clase de método numérico, una red multidimensional es actualizada varias veces al reemplazar el valor en cada punto con una cierta función de los valores en una, fijo pequeño número de puntos vecinos. El conjunto de valores necesarios para actualizar un solo punto de la cuadrícula que se llama punto de la cuadrícula

de *la plantilla* . Por ejemplo, la expresión siguiente se utiliza una plantilla de cinco puntos para actualizar cada elemento X_{ij} de una cuadrícula de dos dimensiones *X* :

$$X_{i,j}^{t+1} = \frac{4X_{i,j}^{t} + X_{i.-1.,j}^{t} + X_{i+1,j}^{t} + X_{i,j-1}^{t} X_{ij+1}^{t}}{8}$$

Esta actualización se aplica varias veces para calcular una secuencia de valores $X_{i,j}^{(1)}$, $X_{i,j}^{(2)}$, Y así sucesivamente. La notación $X_{i,j}^{(t)}$ denota el valor del punto de la malla $X_{i,j}$ en el paso *t* .

Figura 10. Comunicación Local



Foster, I. 1995.

En este grano fino formulación simple, cada tarea encapsula un único elemento de una cuadrícula de dos dimensiones y debe enviar tanto su valor a cuatro vecinos y recibir valores de cuatro vecinos. Sólo los canales utilizados por la tarea se muestran sombreadas.

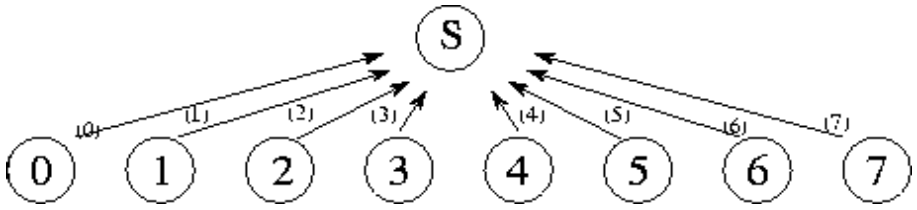
Comunicación Global

Una *comunicación global* es una operación en la que muchas tareas deben participar. Cuando dichas operaciones se aplican, tal vez no sea suficiente la mera identificación de productores individuales o parejas de los consumidores. Este enfoque puede dar lugar a muchas comunicaciones o también pueden restringir las oportunidades de ejecución concurrente. Por ejemplo, considere el problema de llevar a cabo una *reducción paralela de* la operación, que es, una operación que reduce *N* valores distribuidos más de *N* tareas utilizando un operador asociativa conmutativa, como la suma:

$$S = \sum_{i=0}^{N-1} X_i.$$

Se asume una tarea líder que requiere el resultado de S. Desde el punto de vista local, se deben crear canales independientes en donde cada tarea 0,1,2,3,..., n envía su Xi a la tarea líder que debe acumular cada uno de los recibos en la variable S, como se muestra en la figura 11

Figura 11. Comunicación Global



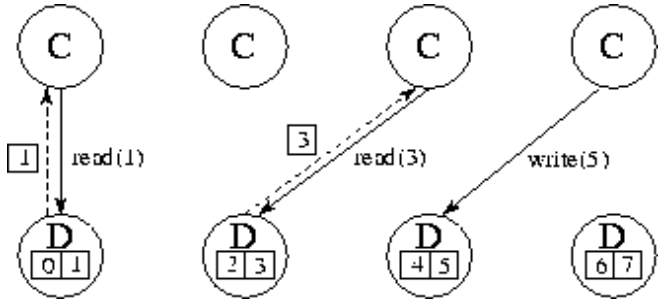
Foster, I. 1995.

Un algoritmo de la suma centralizado que utiliza un administrador de tareas central (S) para sumar N números distribuidos entre N tareas. En este caso, N = 8 , y cada uno de los 8 canales se etiqueta con el número de la etapa en que se utilizan.

La comunicación sincrónica y asincrónica

En la comunicación sincrónica, tanto los productores como los consumidores son conscientes de que las operaciones de comunicación son necesarios, y los productores explícitamente enviar los datos a los consumidores.
En la comunicación asincrónica, las tareas que poseen los datos (los productores) no son capaces de determinar cuándo otras tareas (consumidores) pueden requerir los datos, por lo que los consumidores debe solicitar explícitamente los datos de los productores.

Figura 12. Comunicación síncrona y asíncrona



Foster, I. 1995.

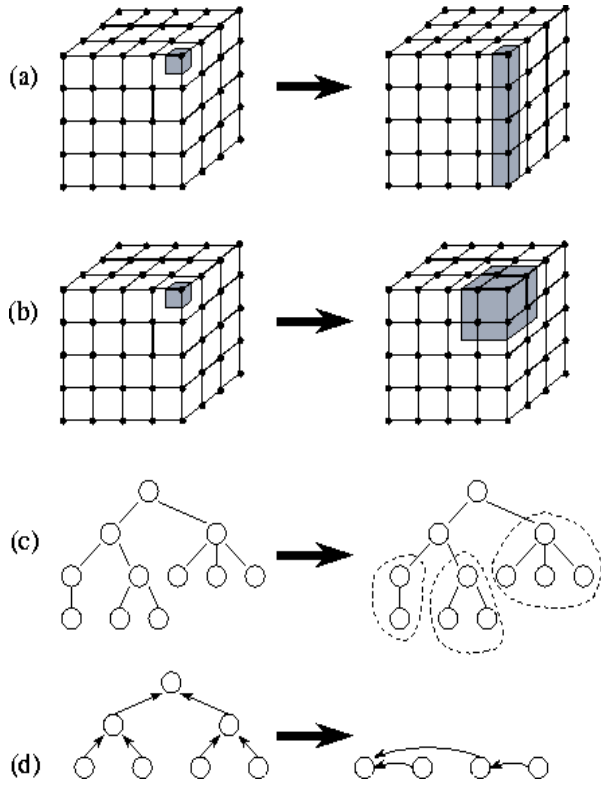
El Uso de tareas por separado los datos para el servicio de lectura y escritura peticiones en una estructura de datos distribuidas. En esta cifra, cuatro tareas de cálculo (C) generan leer y escribir peticiones a ocho elementos de datos distribuidos en cuatro tareas de datos (D). Las líneas continuas representan las solicitudes, las líneas discontinuas representan las respuestas. Un cálculo de tareas y una tarea de datos podría ser colocado en cada uno de cuatro procesadores con el fin de distribuir la computación y los datos de manera equitativa.

2.3.1.3. Aglomeración

En las dos primeras etapas del proceso de diseño, se han dividido el cálculo que se realizará en un conjunto de tareas y se ha introducido la comunicación para proporcionar los datos requeridos por estas tareas. El algoritmo resultante es aún abstracto en el sentido de que no está especializado para la ejecución eficiente en cualquier computador paralelo en particular.

En la tercera etapa, *la aglomeración,* se mueve de lo abstracto hacia lo concreto. Se realiza una revisión de las decisiones tomadas en la fase de partición y la comunicación con el fin de obtener un algoritmo que se ejecutará de manera eficiente en alguna clase de computador paralelo. En particular, consideramos qué es útil para combinar o *aglomerar*, de las tareas identificadas por la fase de particionamiento, a fin de proporcionar un número menor de tareas, cada una de mayor tamaño (Figura 13). También determinar si vale la pena *replicar* datos y / o cálculo.

Figura 13. Aglomeración



Foster, I. 1995.

En la figura 13 Ejemplos de la aglomeración. En (a), el tamaño de las tareas se incrementa mediante la reducción de la dimensión de la descomposición de tres a dos. En (b), las tareas adyacentes se combinan para producir una descomposición de tres dimensiones de mayor granularidad. En (c), sub-estructuras en un "divide y vencerás estructura se unieron. En (d), los nodos de un algoritmo de árbol se combinan.

Tres factores influyen a la hora de agrupar: Granularidad, Flexibilidad y Costos del Software.

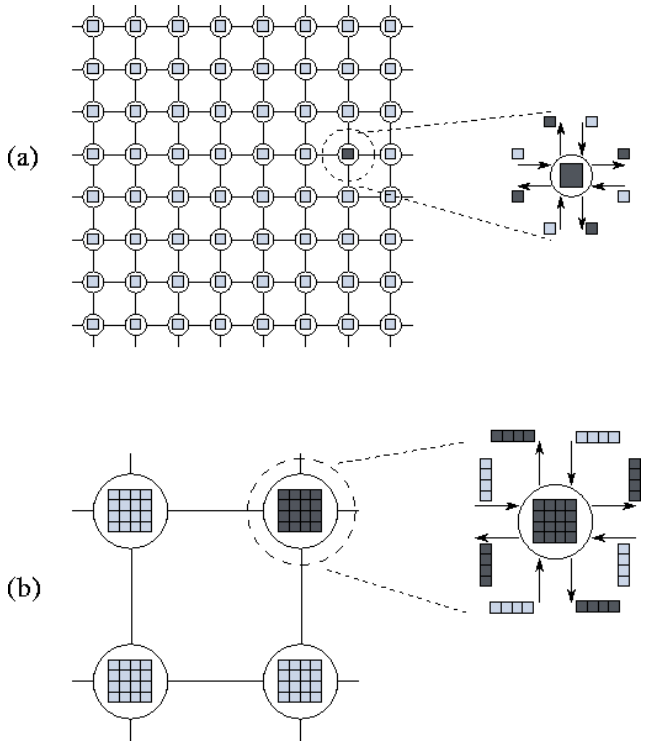
El aumento de granularidad

Uno de los problemas críticos que influyen en el rendimiento en paralelo son los costos de comunicación. En la mayoría de los ordenadores paralelos, se tiene que parar la computación con el fin de enviar y recibir mensajes.

Podemos mejorar el rendimiento al reducir la cantidad de tiempo dedicado a la comunicación. Esta mejora de rendimiento se puede lograr mediante el envío de menos datos y disminuir el número de mensajes, incluso si nos envía la misma cantidad de datos.

Además de los costos de comunicación, es posible que debamos estar preocupados con los costos de creación de la tarea. Por ejemplo, el rendimiento del grano de la búsqueda de un algoritmo, lo que crea una tarea para cada nodo del árbol de búsqueda, es sensible a los costes de creación de la tarea.

Figura 14. Granularidad



Foster, I. 1995.

Efecto de la granularidad aumento en los costos de comunicación en una dimensión finita diferencia problema-dos con un punto de la plantilla de cinco años. La figura muestra de grano de dos particiones dimensiones y gruesos bien de este problema. En todos los casos, una misma tarea se explotó para mostrar sus mensajes salientes (sombreado oscuro) y los mensajes entrantes (gris claro). En (a), un cálculo en un 8 x 8 la red se divide en 8 x 8 = 64 tareas, cada uno responsable de un único punto, mientras que en (b) es partioned el mismo cálculo en 2 x 2 = 4 tareas, cada uno responsable de 16 puntos. En (a), 54 x 4 = 256 comunicaciones se requiere, 4 por tarea, los cuales la transferencia de un total de 256 valores de datos. En (b), sólo 4 x 4 =16 comunicaciones se requiere, y sólo 16 x 4 = 64 valores de los datos se transfieren.

Preservar la flexibilidad

La capacidad de crear un número variable de tareas es fundamental para que un programa sea portable y escalable. Esta flexibilidad también puede ser útil cuando se sintoniza un código para un equipo determinado. La creación de más tareas que procesadores, es que ello proporciona un mayor margen para las estrategias de asignación que equilibrar la carga computacional en los procesadores disponibles.

El número óptimo de las tareas normalmente se determina por una combinación de modelos analíticos y estudios empíricos. La flexibilidad no requiere necesariamente que se creen un gran número de tareas.

Reducción de los costos de Ingeniería de Software

Una consideración adicional, cuando se paralelizan los códigos secuenciales, son los costes de desarrollo relativo asociado con las diferentes estrategias de partición. Desde esta perspectiva, las estrategias más interesantes pueden ser las que evitar grandes cambios de código. Por ejemplo, en un código que funciona en una red multidimensional, puede ser ventajoso preservar las particiones alrededor de una dimensión, si lo permite hacer rutinas existentes para ser reutilizadas sin cambios en un programa paralelo.

Con frecuencia, estamos interesados en el diseño de un algoritmo paralelo que se debe ejecutar como parte de un sistema mayor. En este caso, otro de los temas de ingeniería de software que debe tener en cuenta es la distribución de los datos utilizados por otros componentes del programa.

2.3.1.4. Asignación

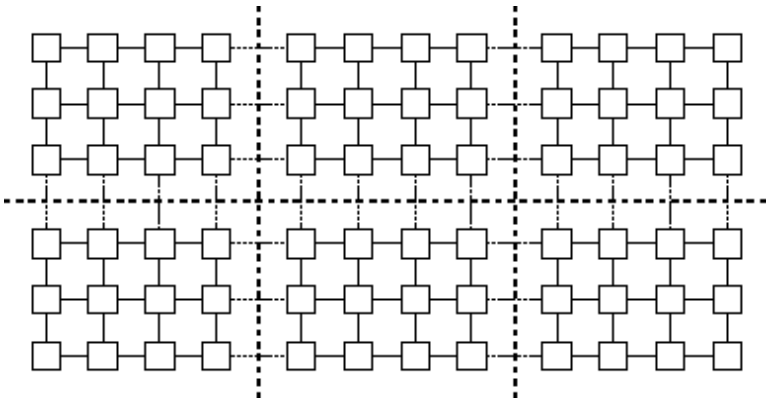
En la cuarta y última etapa del proceso de diseño de algoritmos paralelos, se especifica donde se va a ejecutar cada tarea.
El objetivo en el desarrollo de algoritmos de asignación es normalmente para minimizar el tiempo total de ejecución. Se utilizan dos estrategias para lograr este objetivo:

1. Ponemos las tareas que pueden ejecutar al mismo tiempo en diferentes procesadores, a fin de aumentar la concurrencia.
2. Ponemos las tareas que se comunican con frecuencia en el mismo procesador, con el fin de aumentar la localidad.

Estas dos estrategias a veces causan conflictos, produciendo un desbalanceo de carga. Además, las limitaciones de recursos pueden limitar el número de tareas

asignadas a un procesador.

Figura 15. Asignación



Foster, I. 1995.

Asignación de un problema de red en cada tarea que realiza la misma cantidad de cómputo y sólo se comunica con sus cuatro vecinos. Las líneas de puntos fuertes delinear los límites del procesador. La red de cómputo y asociados se divide para dar a cada procesador de la misma cantidad de cómputo y para reducir al mínimo-procesador de comunicaciones fuera.

Muchos algoritmos desarrollados utilizando técnicas de descomposición de dominio tienen un número fijo de tareas de tamaño igual, comunicación local estructurada y global. En tales casos, una asignación eficiente es sencilla. Esta asignación de tareas minimiza la comunicación entre procesadores, también permite agrupar tareas en el mismo procesador, para dar un total de *P* tareas de grano grueso, una por procesador.

En algoritmos más complejos que se basan en la descomposición de dominio, en los cuales el trabajo por tarea y/o la comunicación no posee una estructura, la

agrupación y la asignación puede no ser evidente para el programador. Por lo tanto, podemos emplear *algoritmos de balanceo de carga* que buscan identificar estrategias eficientes de agrupación y asignación, usualmente mediante técnicas heurísticas. El tiempo requerido para ejecutar estos algoritmos debe sopesarse con el fin de reducir en tiempo de ejecución.

Los problemas más complejos son aquellos en los cuales el número de tareas o la cantidad de comunicación o computación cambia dinámicamente durante el tiempo de ejecución. En ese caso el problema, se pueden usar la *estrategia de balanceo dinámico de carga*, con la cual un algoritmo es ejecutado periódicamente para determinar la nueva aglomeración y asignación.

En algoritmos basados en la descomposición funcional a menudo los cálculos consisten de muchas tareas de corta duración que se coordinan con otras tareas sólo al principio y al final de la ejecución. En este caso, podemos utilizar *la algoritmos de programación de tareas,* que asignan tareas a los procesadores que están inactivos o que pueden llegar a ser inactivos.

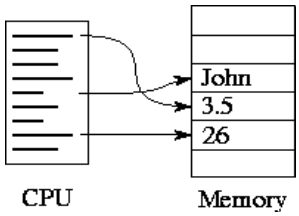
2.4. Computación de Alto Rendimiento

Es una herramienta muy importante en el desarrollo de simulaciones computacionales a problemas complejos, se apoya en tecnologías computacionales como los clusters, supercomputadores o mediante el uso de la computación paralela.

2.4.1. Modelos de Maquinas Paralelas

La rápida penetración de las computadoras en el comercio, la ciencia y la educación debe mucho a la normalización de principios en una sola máquina modelo, el equipo de von Neumann. Un equipo de von Neumann comprende una unidad central de procesamiento (CPU) conectada a una unidad de almacenamiento (memoria) (Figura 16). La CPU ejecuta un programa almacenado que especifica una secuencia de lectura y escritura en la memoria. [21]

Figura 16. Equipo de Von Neumann



Foster, I. 1995.

Una unidad de procesamiento central (CPU) ejecuta un programa que realiza una secuencia de lectura y escritura en una memoria conectada.

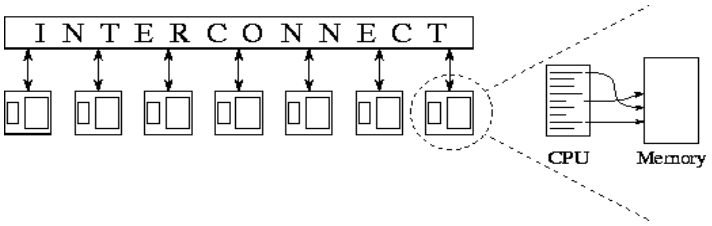
El multicomputador

Como se ilustra en la Figura 16, un multicomputador comprende una serie de Von Neumann computadoras, o *nodos* , vinculados por una *red de interconexión* . Cada computadora ejecuta su propio programa. Este programa puede tener acceso a la memoria local y puede enviar y recibir mensajes a través de la red.

[21] [Fos95]

Los mensajes se utiliza para comunicarse con otros ordenadores o, equivalentemente, a leer y escribir recuerdos remotos. En la red idealizada, el costo de enviar un mensaje entre dos nodos es independiente tanto de localización del nodo y otro tráfico de red, pero no depende de la longitud del mensaje.

Figura 17. Multicomputador



Foster, I. 1995.

Un paralelo modelo de ordenador idealizado. Cada nodo consta de una máquina de von Neumann: una CPU y la memoria. Un nodo puede comunicarse con otros nodos mediante el envío y recepción de mensajes a través de una red de interconexión.

Taxonomía de Flynn

Distingue las arquitecturas multiprocesamiento de cuerdo a las instrucciones y datos. Cada uno de estos criterios puede tomar dos estados:

- Sencillo
- Múltiple

SISD: Single Instruction Single Data
SIMD: Single Instruction Multiple Data
MISD: Multiple Instruction Single Data
MIMD: Multiple Instruction Multiple Data

2.4.2. Procesamiento Distribuido

Un sistema distribuido es un conjunto de computadoras conectadas en red de forma que le da la sensación al usuario de ser una sola computadora. Estos sistemas brindan una serie de ventajas: compartición de recursos, la concurrencia, alta escalabilidad, y tolerancia a fallos. Los sistemas de procesamiento distribuidos brindan una buena relación precio-desempeño y pueden aumentar su tamaño de manera gradual al aumentar la carga de trabajo.

Ventajas

Entre las principales ventajas que presenta el procesamiento distribuido encontramos las siguientes:

- Economía: la relación precio rendimiento es mayor que en los sistemas centralizados sobretodo cuando lo que se buscan son altas prestaciones.
- Velocidad: llega un momento en el que no se puede encontrar un sistema centralizado suficientemente potente, con los sistemas distribuidos siempre se podrá encontrar un sistema más potente uniendo esos mismos nodos. Se han hecho comparativas y los sistemas distribuidos especializados en cómputo han ganado a los mayores mainframes.

- Distribución de máquinas: podemos tener unas máquinas inherentemente distribuidas por el tipo de trabajo que realizan.
- Alta disponibilidad: cuando una máquina falla no tiene que caer todo el sistema sino que este se recupera de las caídas y sigue funcionando con quizás algo menos de velocidad.
- Escalabilidad: puedes empezar un cluster con unas pocas máquinas y según se descubre que la carga es elevada para el sistema, se añaden más máquinas, no hace falta tirar las máquinas antiguas ni inversiones iniciales elevadas para tener máquinas suficientemente potentes.
- Comunicación: los ordenadores necesariamente están comunicados, para el correcto y eficaz funcionamiento del cluster se crean unas nuevas funcionalidad es avanzadas de comunicación. Estas nuevas primitivas de comunicación pueden ser usadas por los programas y por los usuarios para mejorar sus comunicaciones con otras máquinas.
- Sistema de ficheros con raíz única: este sistema de ficheros hace que la administración sea más sencilla (no hay que administrar varios discos independientemente) y deja a cargo del sistema varias de las tareas.
- Capacidad de comunicación de procesos y de intercambio de datos universal: podemos enviar señales a cualquier proceso del cluster, así mismo podemos hacer trabajo conjunto con cualquier proceso e intercambiar con los datos, por lo tanto podríamos tener a todos los procesos trabajando en un mismo trabajo.

Desventajas

La principal desventaja de estos sistemas es la complejidad que implica su creación. Básicamente se tienen todos los problemas que se puedan tener en un nodo particular pero escalados. Existen varias desventajas asociadas a las ventajas anteriormente reseñadas:

- Alta disponibilidad: podemos conseguir alta disponibilidad pues al tener varios nodos independientes, hay muchas menos posibilidades de que caigan todos a la vez. Pero esto por sí sólo no nos da alta disponibilidad. Tenemos que implantar los mecanismos necesarios para que cuando una máquina caiga, se sigan dando todos los servicios.
- Escalabilidad: el problema es que un mayor número de nodos suele implicar más comunicación, por lo que tenemos que diseñar un sistema lo más escalable posible.
- Comunicación: un cluster tiene más necesidades de comunicación que los sistemas normales por lo tanto tenemos que crear nuevos métodos de comunicación lo más eficientes posibles.
- Sistemas de ficheros con raíz única: tenemos que independizar los sistemas de ficheros distintos de cada uno de los nodos para crear un sistema de ficheros general.
- Capacidad de comunicación de procesos y de intercambio de datos universal: para conseguir este objetivo necesitamos una forma de distinguir unívocamente cada proceso del cluster. La forma más sencilla es dando a cada proceso un PID único, que llamaremos CPID (cluster process ID). Este CPID podría estar formado por el número de nodo y el número de proceso dentro de ese nodo. Una vez podemos direccionar con que proceso queremos comunicarnos, para enviar señales necesitaremos un sencillo mecanismo de comunicación y seguramente el mismo sistema operativo en el otro extremo que entienda las señales. Para compartir datos, podemos enviarlos por la red o podemos crear memoria compartida a lo largo del cluster.

2.4.3. Clusters de Computadoras

Un cluster es un conjunto de máquinas unidas por una red de comunicación trabajando por un objetivo conjunto. Es la variación de bajo precio de un multiprocesador masivamente paralelo (miles de procesadores, memoria distribuida, red de baja latencia), con las siguientes diferencias: cada nodo es una máquina quizás sin algo del hardware (monitor, teclado, ratón, etc.), el nodo podría ser SMP o PC. Los nodos se conectan por una red de bajo precio como Ethernet o ATM aunque en clusters comerciales se pueden usar tecnologías de red propias. El interfaz de red no está muy acoplado al bus I/O. Todos los nodos tienen disco local.

Características generales

- Un cluster consta de 2 o más nodos. Los nodos necesitan estar conectados para llevar a cabo su misión.
- Los nodos de un cluster están conectados entre sí por un canal de comunicación funcional.
- Los clusters necesitan software especializado. Existen varios tipos de software que pueden conformar un cluster, software a nivel de aplicación y software a nivel de sistema.
- Dependiendo del tipo de software, el sistema puede estar más o menos acoplado. Entendemos por acoplamiento del software a la integración que tengan todos los elementos software que existan en cada nodo.
- Todos los elementos del cluster trabajan para cumplir una funcionalidad conjunta, sea esta la que sea. Es la funcionalidad la que caracteriza el sistema.

Tipos de clusters

Existen tres tipos fundamentales de clusters: clusters de alto rendimiento, clusters de alta disponibilidad y clusters de alta confiabilidad.

Clusters de alto rendimiento

Los clusters de alto rendimiento han sido creados para compartir el recurso más valioso de un ordenador, es decir, el tiempo de proceso. Generalmente se utilizan en ambientes científicos, o en grandes empresas donde se utilizan para la compilación o renderización. Cualquier operación que necesite altos tiempos de CPU y millones de operaciones puede ser implementada en un cluster de alto rendimiento, siempre que se encuentre un algoritmo que sea paralelizable.

La misión de este tipo de clusters es mejorar el rendimiento en la obtención de la solución de un problema. Esto supone que cualquier cluster que haga que el rendimiento del sistema general aumente respecto al de uno de los nodos individuales puede ser considerado cluster de alto rendimiento. Generalmente, los problemas que se le plantean a un ordenador, suelen ser de carácter computacional.

Clusters de alta disponibilidad

Los clusters de alta disponibilidad son bastante ortogonales en lo que se refieren a funcionalidad con respecto a un cluster de alto rendimiento. Los clusters de alta disponibilidad son clusters donde la principal funcionalidad es estar controlando y actuando para que un servicio o varios se encuentren activos durante el máximo

periodo de tiempo posible. En estos casos se puede comprobar como la monitorización de otros es parte de la colaboración entre los nodos del cluster.

Los clusters de alta disponibilidad han sido diseñados para que proporcionen la máxima disponibilidad sobre los servicios que presenta el cluster. Este tipo de clusters son la competencia que abarata los sistemas redundantes, de manera que ofrecen una serie de servicios durante el mayor tiempo posible. Para poder dar estos servicios, los clusters de este tipo se implementan en base a tres factores:
confiabilidad, disponibilidad y dotación de servicio.

Clusters de alta confiabilidad

Estos clusters tratan de aportar la máxima confiabilidad en un entorno en el cual se necesite saber que el sistema se va a comportar de una manera determinada.
Este tipo de clusters son los más difíciles de implementar, ya que generalmente se basan en no solamente conceder servicios de alta disponibilidad, sino en ofrecer un entorno de sistema altamente confiable. Esto implica en sí mismo mucha sobrecarga en el sistema, son también clusters muy acoplados.

Dar a un cluster SSI capacidad de alta confiabilidad implica gastar los recursos necesarios para evitar que aplicaciones caigan.

Generalmente este tipo de clusters suele ser utilizado para entornos de tipo empresarial y esta funcionalidad solamente puede ser efectuada por hardware especializado. No existe ninguno de estos clusters implementados como software de manera eficiente en tiempo real. Esto se debe a limitaciones de la latencia de la red, así como a la complejidad de mantener los estados. Estos clusters deberían ser una mezcla de clusters de alto rendimiento y alta disponibilidad mejorados.

Necesitarían características de cluster SSI, tener un único reloj de sistema conjunto y otras acciones más. Dada la naturaleza asíncrona actual en el campo de los clusters, este tipo de clusters aún será difícil de implementar hasta que no se abaraten las técnicas de comunicación utilizadas en entornos que actualmente consideramos inalcanzables para la mayoría de las organizaciones o con no suficiente coste/rendimiento.

2.4.4.Topologías de Redes Paralelas

La topología se refiere a la forma como se enlazan los procesadores en un computador en paralelo, su importancia radica en que permite minimizar el acceso de memoria. Los criterios para evaluar el diseño son:

Diámetro de la red. Es la mayor distancia entre dos nodos, entre menor se el diámetro, menor es el tiempo de comunicación.

La conectividad de la red. Permite reducir el tiempo de comunicación al poder viajar por caminos alternos, evitando o reduciendo la congestión entre los nodos de la red.

La flexibilidad. Permite ejecutar una amplia variedad de algoritmos, en donde la interconexión puede variar durante la ejecución de los programas.

El retraso en la comunicación. Requerido por algunas tareas en muchos algoritmos como computación de productos internos,multiplicación entre una matriz y un vector, etc.

Tipos de Topologías

Arreglo de procesadores lineal. Permite comunicación bidireccionales entre cada uno de los procesadores que forman un elemento del vector.

Anillo. Cada nodo se conecta al siguiente y el último al primero formando un anillo.

Árbol. Existe un nodo llamado raíz, en donde cada nodo i se conecta por un solo camino a otro nodo en grupos de dos hasta llegar al nodo raíz.

Matriz. Arreglos de procesadores conectados en d dimensiones.

Hipercubo. Los nodos son agrupados en dos cubos desplazados en un cuarto eje dimensional, utilizando 32 conexiones entre ellos 33.

2.4.5. Modelos de Programación Paralela

Un modelo de programación paralela o paradigma es un conjunto de tecnologías de software que permiten expresar algoritmos paralelos para implantar aplicaciones en la arquitectura adecuada.

Un modelo de programación paralela incluye distintas áreas:

- Aplicaciones
- Lenguajes de programación
- Compiladores
- Bibliotecas

- Sistemas de comunicación
- Dispositivos de I/O paralelos

Un modelo de programación paralela es implantado de distintas formas:

- Como bibliotecas invocadas desde programas tradicionales
- Como extensiones de lenguajes de programación
- Como modelos de ejecución completamente nuevos

Una primera categorización de estos modelos se realiza de acuerdo al manejo de la memoria:

- Memoria compartida (shared memory)
- Memoria distribuida (distributed memory)
- Memoria compartida distribuida (distributed shared memory)

El paralelismo de un programa lo podemos obtener de distintas fuentes:

- Paralelismo de Bits
- Paralelismo a nivel de instrucciones
- Paralelismo a nivel de datos
- Paralelismo funcional

2.4.6. MPI (Message Passing Interface)

MPI es un Message Passing Interface estándar definido por un grupo compuesto por unas 60 personas de 40 organizaciones de los Estados Unidos y Europa,

incluyendo proveedores, así como investigadores. Fue el primer intento de crear una "norma de consenso" para las bibliotecas de paso de mensajes. MPI está disponible en una amplia variedad de plataformas, que van desde sistemas masivamente paralelos a las redes de estaciones de trabajo.
El documento que define MPI es "MPI: Un paso de mensajes estándar ", escrito por el Foro de Mensajes Passing Interface[22]

Objetivos de MPI [23]

El objetivo de MPI es desarrollar un estándar para ser ampliamente usado que permita escribir programas usando pase de mensajes. Por lo tanto, la interfase debería establecer un estándar práctico, portable, eficiente y flexible.

A continuación se presentan los objetivos de MPI:

- Diseñar una API (Application Programming Interface).
- Hacer eficiente la comunicación.
- Permitir que las aplicaciones puedan usarse en ambientes heterogéneos.
- Soportar conexiones de la interfase con C y Fortran 77, con una semántica independiente del lenguaje.
- Proveer una interfase de comunicación confiable.
- No diferir significativamente de algunas implementaciones tales como PVM, p4, NX, express, etc., extendiendo la flexibilidad de éstas.
- Definir una interfase implementable en plataformas de diferentes proveedores sin tener que hacer cambios significativos.

[22] http://www.netlib.org/mpi
[23] [Seo03]

Características

Generales:

- Los comunicadores combinan procesamiento en contexto y por grupo para seguridad de los mensajes.
- Seguridad en la manipulación de aplicaciones con hilos.

Manejo de ambiente. MPI incluye definiciones para:

- Temporizadores y sincronizadores.
- Inicializar y finalizar.
- Control de errores.
- Interacción con el ambiente de ejecución.

Comunicación punto a punto:

- Heterogeneidad para buffers estructurados y tipos de datos derivados.
- Varios modos de comunicación: Normal, con y sin bloqueo; Síncrono; Listo, para permitir acceso a protocolos rápidos.; Retardados, utilizando buffers.

Comunicaciones colectivas:

- Capacidad de manipulación de operaciones colectivas con operaciones propias o definidas por el usuario.
- Gran número de rutinas para el movimiento de datos.
- Los subgrupos pueden definirse directamente o por la topología.

Topologías por procesos:

- Soporte incluido para topologías virtuales para procesos (mallas y grafos).

Caracterización de la Interfase:

- Se permite al usuario interceptar llamadas MPI para instalar sus propias herramientas.

MPI Básico [24]

La especificación completa de MPI consiste en cerca de 129 llamadas. Sin embargo, un principio programador de MPI puede llegar a funcionar con muy pocos de ellos (de 6 a 24). Todo lo que es realmente necesario es una forma de que los procesos de intercambio de datos, es decir, ser capaz de enviar y recibir mensajes.

Las siguientes son las funciones básicas que se utilizan para construir la mayoría de los programas MPI:

- Todos los MPI / programas en C debe incluir un archivo de encabezado mpi.h.
- Todos los programas MPI MPI INT debe llamar como la primera convocatoria de MPI, para inicializar ellos mismos.
- La mayoría de los programas MPI llamada MPI COMM SIZE para obtener el número de procesos que se ejecutan.

[24] [HH]

- La mayoría de los programas MPI llamada MPI COMM RANK para determinar su rango, que es un número entre 0 y el tamaño de 1.
- El proceso condicional y paso de mensajes en general pueda tener lugar. Por ejemplo, usando las llamadas MPI SEND y RECV MPI.
- Todos los programas MPI debe llamar MPI concluya tal y como la última llamada a MPI la colección de rutina.

Así que podemos escribir una serie de útiles programas MPI utilizando sólo los siguientes seis llamadas MPI INIT, MPI COMM SIZE, MPI COMM RANK, ENVIAR MPI, RECV MPI, MPI FINALIZAR.

El siguiente es uno de los sencillos MPI / programas en C++ que hace que todos los procesadores impriman "Hello, I am *x* of *y*".

```
#include <iostream>
#include <mpi.h>

using namespace std;

int main(int argc, char* argv[]) {
   MPI::Init(argc, argv);
   cout << "Hello, I am " << MPI::COMM_WORLD.Get_rank();
   cout << " of " << MPI::COMM_WORLD.Get_size();

   MPI::Finalize();
   return 0;
}
```

En el programa de ejemplo, el proceso principal (rango = 0) envía un mensaje que consiste en los caracteres "Hola, mundo" a todos los demás procesos (clasificación> 0).
Los otros procesos simplemente recibir este mensaje e imprimirlo.
Se puede compilar con alguna de estas dos instrucciones:

g++ -o executable hello.cpp -lmpi

mpiCC -o executable hello.cpp

Se ejecuta con el comando: mpirun -np 4 executable, la salida sería:

Hello, I am 3 of 4
Hello, I am 1 of 4
Hello, I am 2 of 4
Hello, I am 0 of 4

2.4.7. Procesamiento Distribuido con C++

Estándares para la programación paralela en C++

El lenguaje C++ no incluye ninguna primitiva para poder realizar la implementación de aplicaciones paralelas. No existe ninguna manera a través del lenguaje C++ para especificar que dos o más instrucciones deberían ser ejecutadas de forma paralela.

Aunque existen versiones especiales de C++ que implementan la programación paralela, vamos a presentar una serie de métodos sobre cómo la programación paralela puede ser implementada usando el estándar de ISO para C++.

La librería que da soporte a la programación paralela es la más flexible. Las librerías del sistema y las librerías del usuario pueden ser empleadas para soportar el paralelismo en C++. Las librerías del sistema son aquellas librerías que son proporcionadas por el propio sistema operativo. Por ejemplo, la librería POSIX (Portable Operating System Interface) es un conjunto de llamadas del sistema que puede ser usado conjuntamente con C++ para soportar la programación paralela.

Estas librerías forman parte de una nueva especificación de UNIX.

El estándar MPI

MPI es la implementación estándar para llevar a cabo la comunicación basándose en el intercambio de mensajes. MPI fue desarrollado para su ejecución tanto en máquinas paralelas como en clusters de estaciones de trabajo. Para trabajar con MPI utilizamos la implementación MPICH. MPICH es una implementación de MPI libre y portable. MPICH proporciona al programador de C++ un conjunto de APIs y librerías que soportan programación paralela. MPI es especialmente útil para la programación SPMD (Single Program Multiple Data) y MPMD (Multiple Program Multiple Data).

2.4.6. Métricas de Rendimiento

Para poder evaluar el desempeño de un sistema de computación y así poder compararlo respecto a otro necesitamos definir y medir su rendimiento. Pero, ¿Qué queremos decir con rendimiento?, ¿En base a qué parámetros podemos expresar o medir el rendimiento?

¿Cómo podemos establecer un mecanismo que nos permita comparar dos sistemas de computación?

Para poder cuantificar el rendimiento, necesitamos determinar los factores que influyen en el desempeño del equipo de cómputo y así definir una expresión que caracterice este rendimiento. Se denomina medida al valor obtenido mediante un instrumento de medición confiable. Una medida “proporciona una indicación cuantitativa de extensión, cantidad, dimensiones, capacidad y tamaño de algunos atributos de un proceso o producto”

El desempeño de un computador puede tener diferentes medidas de elección para diferentes usuarios. Para un usuario individual que está ejecutando un único programa, el computador con mayor rendimiento es aquel que complete la ejecución de su programa en menor tiempo. Sin embargo, para el administrador de un centro de cómputos, que tiene múltiples tareas que realizar a la vez, la de mayor rendimiento es la que le realice más tareas en menor tiempo. Como elemento común, sin embargo, se evidencia que la medida del rendimiento del computador es el tiempo. El computador que ejecute los programas en menor tiempo es la que tiene mejor rendimiento.

Un algoritmo secuencial es evaluado por su tiempo de ejecución como función del tamaño del problema. El comportamiento asintótico del tiempo de ejecución es

idéntico en cualquier plataforma secuencial. En cambio, el tiempo de ejecución de un programa paralelo depende del tamaño del problema, del número de procesadores y de ciertos parámetros de comunicación de la plataforma. Es por ello que los algoritmos paralelos deben ser evaluados y analizados teniendo en cuenta también la plataforma Además, en el mundo paralelo, adicional al tiempo de ejecución como medida de rendimiento también encontramos otras métricas como la aceleración (speedup), eficiencia (efficiency) , etc.

Definiciones:

Rendimiento

No es sencillo definir qué es rendimiento. Por ejemplo la tarea de los ingenieros de software es diseñar e implementar programas que satisfagan los requerimientos del usuario relacionados con correctivo y rendimiento. Si ejecutamos un programa en dos computadores distintos, ustedes dirían que el computador más rápido es aquel que termina primero la ejecución del programa. Si tienen un centro de computación que dispone de dos plataformas computacionales distintas que ejecutan trabajos de muchos usuarios, dirían que el sistema más rápido es aquel que ejecuta y completa más trabajos por día. Como usuario, a usted le interesa reducir el tiempo de respuesta y el tiempo de ejecución de su programa. Por lo tanto su percepción de rendimiento puede cambiar dependiendo de la situación.

Escalabilidad

Otro término importante es el de Escalabilidad. Este término se refiere al estudio del cambio en las medidas de rendimiento de un sistema cuando una o varias características del sistema son variadas. Una de las características que

generalmente se cambia es el número de procesadores. Otra característica relacionada con paralelismo que puede ser modificada es la granularidad del problema.

Granularidad

Granularidad consiste en la cantidad de cómputo con relación a la comunicación. En Granularidad Fina o Fine-grained, las tareas individuales son relativamente pequeñas en término de tiempo de ejecución. La comunicación entre los procesadores es frecuente.

En cambio en Granularidad gruesa o Coarse-grained, la comunicación entre los procesadores es poco frecuente y se realiza después de largos periodos de ejecución.

Métricas:

Aceleración (Speedup)

La medición de rendimiento en ambientes paralelos es más complejo por nuestro deseo de conocer cuánto más rápido ejecuta una aplicación en un computador paralelo. Es decir, nos interesa conocer cuál es el beneficio obtenido cuando usamos paralelismo y cuál es la aceleración que resulta por el uso de dicho paralelismo.

La aceleración puede ser definida como

$$S = \frac{Tiempo\ \ Ejecución\ \ Secuencial}{Tiempo\ \ Ejecución\ \ Paralelo}$$

Existen diversas formas de expresar la aceleración que dependen de la forma en que se defina lo que es el Tiempo de Ejecución Secuencial y Paralelo.

Aceleración Relativa. El Tiempo de Ejecución Secuencial usado es el tiempo total de ejecución del programa paralelo cuando éste es ejecutado sobre un único procesador del computador paralelo. Por lo tanto la Aceleración Relativa de un programa paralelo Q cuando se resuelve una instancia I de tamaño n usando P procesadores es:

$$Aceleración\ \mathrm{Re}lativa(I,P) = \frac{T(I,1)}{T(I,P)}$$

Aceleración Real. El Tiempo de Ejecución Secuencial usado es el tiempo del mejor programa secuencial para resolver el problema.

$$Aceleración\ Real(I,P) = \frac{T^*}{T(I,P)}$$

Formalmente la aceleración es la relación entre el tiempo de ejecución sobre un procesador secuencial y el tiempo de ejecución en múltiples procesadores.

La ley de Amdahl

Un aspecto importante a considerar es que todo programa paralelo tiene una parte secuencial que eventualmente limita la aceleración que se puede alcanzar en una plataforma paralela. Por ejemplo, si el componente secuencial de un algoritmo es 1/s de su tiempo de ejecución, entonces la aceleración máxima posible que puede alcanzar en el computador paralelo es s.

El tiempo para realizar el cómputo con P procesadores es

$$T_P = st_s + \frac{(1-s)t_s}{P}$$

donde ts es el tiempo de ejecución secuencial

La aceleración puede ser redefinida de la siguiente forma:

$$Aceleración = \frac{1}{s + \frac{(1-s)}{P}} = \frac{P}{Ps + 1 - s} = \frac{P}{1 + (P-1)s}$$

$$\frac{P}{1+(P-1)s} \to \frac{1}{s} \quad cuando\ P \to \infty$$

Según la ley de Amdahl, si un programa tiene un 5% de componente secuencial entonces la aceleración máxima que se puede alcanzar es de 20.

La ley de Amdahl tiene varias implicaciones:

- Para una carga dada, la aceleración máxima tiene una cota superior de 1/s. Al aumentar s, la aceleración decrecerá proporcionalmente.

- Para alcanzar buenas aceleraciones, es importante reducir s

Ley de Gustafson

La ley de Gustafson (1988) viene a compensar el pesimismo dejado por la ley de Amdahl.

Ésta se refería a problemas con volumen de cálculo fijo en que se aumenta el número de procesadores. Sin embargo, en la práctica, el volumen del problema no es independiente del número de procesadores, ya que con mayor número de procesadores se pueden abordar problemas de mayores dimensiones. Por ello, la ley de Gustafson se refiere al crecimiento del volumen de cálculo necesario para resolver un problema. Como veremos a continuación, en la mayoría de los casos, cuando el volumen del problema crece, lo hace sólo en su parte paralela, no en su parte secuencial. Ello hace que el cuello de botella secuencial tienda a cero cuando el volumen del problema aumenta.

La razón de que cierto tipo de problemas adquieran gran volumen de cálculo es la disminución del tamaño de la malla de cálculo o también el aumento la extensión espacio-temporal del problema. Esto hace que el número de puntos aumente de forma cúbica respecto al grado de disminución en la malla, si el problema es tridimensional. Hay muchos problemas en que, además de las tres dimensiones del espacio, también interviene el tiempo, por lo que el aumento del volumen de cálculo es todavía mayor.

Evidentemente, esto afecta, en general, a la parte paralelizable del problema y no a su parte secuencial, o al menos no en la misma medida. Si suponemos que el número de procesadores crece indefinidamente de la misma forma que las dimensiones del problema tendremos que siendo s y p, respectivamente, las partes secuencial y paralela del problema antes de ser aumentado en relación al número del procesadores (por ejemplo, incrementando el número de puntos de la malla).

Con estas premisas, podremos calcular ahora la ganancia de velocidad para esta nueva situación (la parte paralela del problema ha crecido en la misma proporción que el número de procesadores). Para calcular la ganancia de velocidad supondremos que el tiempo que se tardaría en ejecutar el programa (ya incrementado) en un monoprocesador es:

$$T(1) = s + Np$$

y en un sistema paralelo sería:

$$t(N) = s + p$$

Por tanto, la ganancia en velocidad vendrá dada por:

$$S = t(1) / t(N) = s + Np / s+p$$

que, teniendo en cuenta la definición de la fracción no paralelizable, se podrá escribir como:

$$S = f + N(1-f) = N - (N-1)f$$

Eficiencia

Podemos definir la eficiencia como el porcentaje de tiempo empleado en proceso efectivo

$$E = \frac{S}{P}$$

Escalabilidad

Un sistema es escalable si mantiene constante la eficiencia al aumentar el número de procesadores aumentando también el tamaño del problema

Cómo medimos el tiempo de ejecución de un programa paralelo?

En el caso de un programa secuencial, el tiempo de ejecución es el tiempo que transcurre desde que se inicia la ejecución hasta que finaliza. En el caso de un programa paralelo es el tiempo transcurrido desde que comienza su ejecución hasta el momento en que el último procesador finaliza su ejecución (TP).

Capítulo 3

Metodología de Desarrollo

En este capítulo se sustentan los pasos realizados para la ejecución de la investigación, los conceptos requeridos, se especifica las herramientas que se utilizaron para implementar el proyecto.

Luego se justifica las características de las imágenes médicas objetivo, las cuales fueron utilizadas para segmentar con el algoritmo paralelo desarrollado.

3.1. Estructura de la Metodología

La metodología de desarrollo del proyecto se fundamenta en la estructura de gestión de proyectos, y la estrategia de desarrollo se basa en el prototipado evolutivo ya que es la más adecuada para la realización del proyecto.

La gestión de proyectos tiene como finalidad principal la planificación, el seguimiento y control de las actividades y de los recursos humanos y materiales que intervienen en el desarrollo de un Sistema de Información. Como consecuencia de este control es posible conocer en todo momento qué problemas se producen y resolverlos o paliarlos de manera inmediata.

Mientras que la metodología especifica procesos, actividades, tareas y productos a obtener, la estrategia de desarrollo es el enfoque a utilizar para establecer cómo debe organizarse el proyecto.

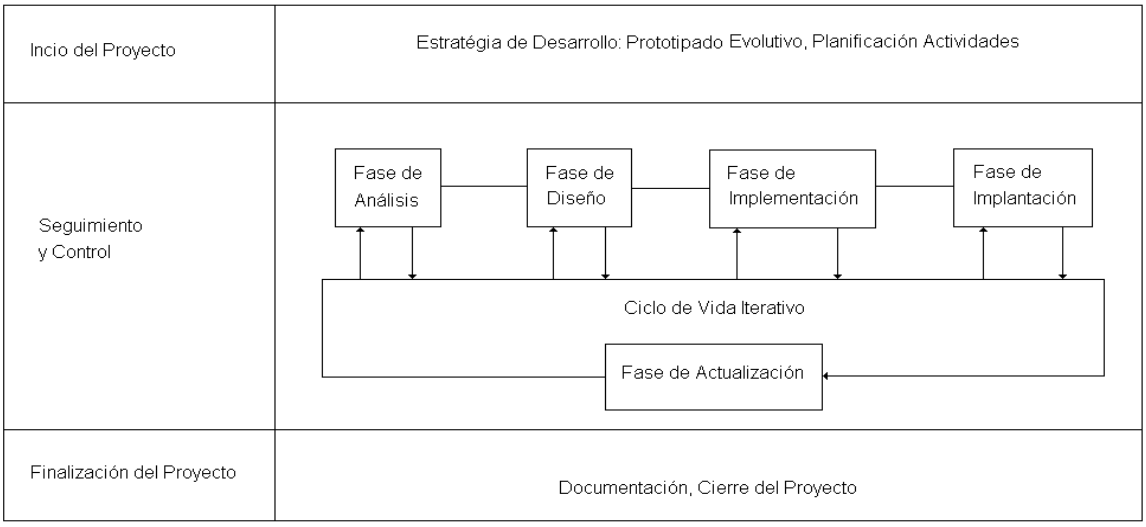
Grupo de actividades de la interfaz de gestión de proyectos:

- Actividades de Inicio del Proyecto: Planificación del proyecto.

- Actividades de Seguimiento y Control. Se realizan durante los procesos de Análisis, Diseño, Implementación, Implantación y Actualización del Proyecto,para vigilar el correcto desarrollo de las actividades y tareas establecidas en la planificación.

- Actividades de Finalización del Proyecto. Al concluir el proyecto se realizan las tareas propias de Cierre del Proyecto y Registro de la Documentación.

Figura 18. Estructura de la Metodología de Desarrollo.



Autora Documento

3.2. Estudio y Análisis Teórico del Problema

La investigación de los fundamentos teóricos y el estado de arte fue esencial ya que no se tenia conocimiento sobre la computación de alto rendimiento y fue necesario el entrenamiento en las herramientas utilizadas desde el inicio.

Los artículos de investigación sobre el tema fueron de gran ayuda, ya que de ellos se comprendió a fondo aspectos necesarios para la solución correcta del problema, como la forma de identificar las regiones, la cual ayuda a que el desbalanceo de carga sea menor.

3.2.1. Preprocesamiento de la imagen a segmentar

Antes de realizar la segmentación de la imagen con el algoritmo paralelo implementado, las imágenes fueron convertidas a la escala de grises.

Tipos de imágenes utilizadas

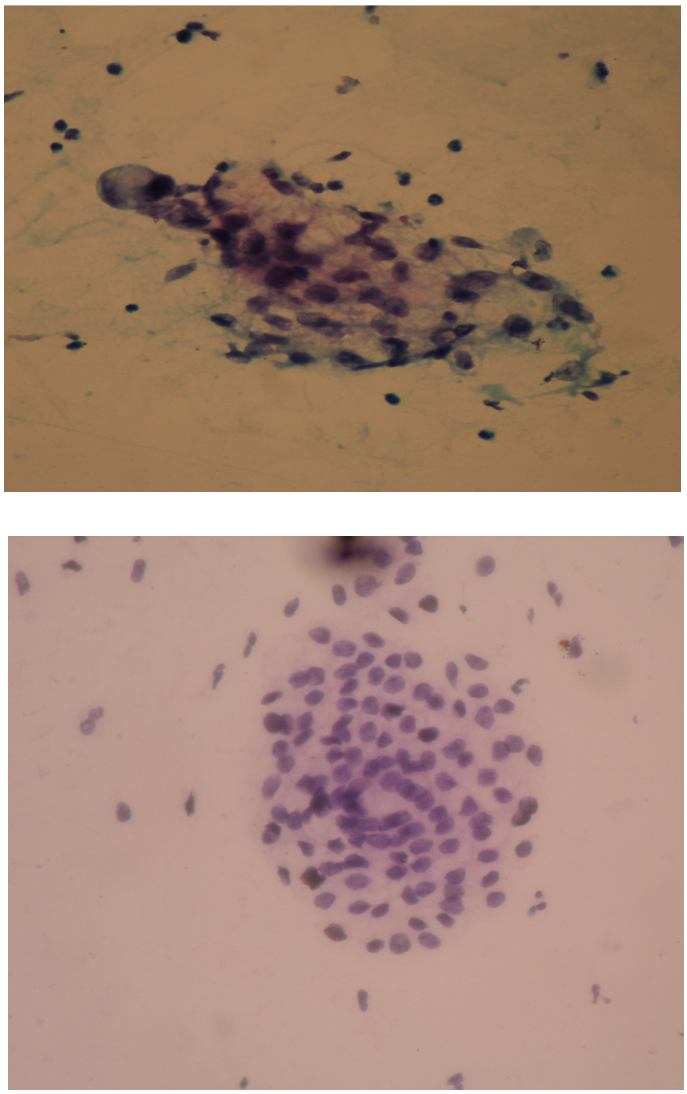
Las imágenes utilizadas para hacer las pruebas del algoritmo fueron tomadas de una investigación anterior.[25]

El criterio de selección se realizó por la cantidad de objetos que deben ser segmentados.

[25] [Mar07b] p.57-59.

Las imágenes son en formato TIFF, tienen una resolución de 1200x1200 píxeles y tienen un tamaño promedio de 1.7 MB.

Figura 19. Ejemplos de Imágenes a Segmentar



Martinez, V. [Mar04]

3.3. Diseño del Algoritmo

El diseño se realizó siguiendo la metodología de Ian Foster para algoritmos paralelos: particionamiento, comunicación, aglomeración y asignación. Adicionalmente fue guiado por investigaciones relacionadas con el tema, mencionadas en el estado del arte (Capitulo 1).

Un paso clave para obtener un buen resultado en la segmentación, es la elección del criterio de homogeneidad para evaluar las regiones y tomar la mejor decisión al momento de realizar el Split and Merge. Tanto para la evaluación del criterio de homogeneidad en la fase de fusión o división es el mismo, se define a continuación:

3.3.1. Criterio de Homogeneidad

El criterio de homogeneidad de la región está dado por los niveles de gris de los píxeles que la componen, estableciendo un criterio de uniformidad. V Let $X_{i,\ i=1,2,3,...n}$ los niveles de gris de los píxeles que conforman una región R. El valor promedio de niveles de gris en la región R, es la media aritmética de los niveles de gris de los píxeles: [26]

$$\bar{X}_R = \frac{\sum_{i=1}^{n} X_i}{n} \qquad (1)$$

[26] [ARR01]

Una región R es uniforme en niveles de gris, si para cada sub-región $r_i \in R$, se satisface que la relación entre el mínimo de los valores medios de niveles de gris de la región R y una sub-región r_i y el máximo de estos valores es menor o igual a 0,95. Es decir,

$$\frac{Min\left(\bar{X}_R, \bar{X}_{ri}\right)}{Max\left(\bar{X}_R, \bar{X}_{ri}\right)} \leq 0{,}95 \;\forall\; ri, i = 1, \ldots 4 \qquad (2)$$

La región es homogénea en los niveles de gris si cumple con el criterio de uniformidad.

Calculo de los niveles de gris en las regiones

La textura se midió utilizando el calculo de la diferencia estadística de los valores de gris. Los criterios de homogeneidad utilizados en la región de fusión dependen de las estadísticas locales.

3.4. Implementación del Algoritmo

Para evaluar el rendimiento del algoritmo Split & Merge, se realizaron tres prototipos, todos iguales al diseño propuesto, a excepción de que en cada prototipo se variaron las características de la identificación de regiones, el sistema de Comunicaciones y los algoritmos de balance de carga. Con el fin de observar la variabilidad del algoritmo respecto a estas tres características.

A continuación se presentan las variaciones de cada prototipo:

3.4.1. Prototipo I

- Identificación de regiones ascendente
- Comunicación Global
- Sin Algoritmo de Balance de Carga

3.4.2. Prototipo II

3. Identificación de regiones cíclico y aleatorio
4. Comunicación Global
5. Con Algoritmo Balance de Carga Dinámico

3.4.3. Prototipo III

- Identificación de regiones cíclico y aleatorio
- Comunicación Topología Grafo
- Con Algoritmo Balance de Carga Estático y Dinámico

3.4.4. Especificaciones de Software

Se ha elegido C++ como lenguaje de programación, bajo el sistema operativo Linux, gcc es el lenguaje estándar diseñado para Linux y para la comunicación por paso de mensajes OpenMPI.

Se escogió trabajar sobre esta plataforma por la simplicidad de manejo de un Cluster Linux y la programación en gcc, brindando un ambiente propicio para la computación paralela para este tipo de algoritmo que no necesita de un ambiente mas robusto.

3.4.5. Especificaciones de Hardware

Las pruebas finales del algoritmo fueron realizadas en un Cluster personal con las siguientes especificaciones:

Nodo1:
HP Pavilion 1.81 GHz, 2 GB Memoria RAM, Dual Core
Procesador 0: AMD Turion

Nodo2:
HP Pavilion 1.81 GHz, 0.5 GB Memoria RAM, Dual Core
Procesador 1: AMD Turion

Nodo3:
HP Pavilion 1.60 GHz, 1 GB Memoria RAM
Procesador 0: AMD Dual Core

Nodo4:
HP Pavilion 1.60 GHz, 1 GB Memoria RAM,
Procesador 1: AMD Dual Core

Alternativamente también se utilizaron los clusters LEAC y GIIB de la Plataforma de calculo científico de la Universidad Industrial de Santander, ubicado en el Centic - 4to piso, y en el Grupo de Investigación en Ingeniería Biomédica. A continuación los detalles de los clusters.

Cluster LEAC:

Nodos disponibles para el calculo: 3

Nodo1: *nodo01-leac.uis.edu.co*
DELL Powerdege 2850, 3.60 GHz Intel Xeon, 2 GB Memoria RAM
Tarjeta de video ATI Radeon 7000 y 4 cores

Nodo2: *nodo02-leac.uis.edu.co*
DELL Powerdege 2950, 2.50GHz Intel Xeon, 2 GB Memoria RAM
Tarjeta de video ATI ES1000 y 4 cores

Nodo3: *nodo03-leac.uis.edu.co*
DELL Precision T3400, 2.66 GHz Intel Core 2 Quad, 2 GB de Memoria RAM
Tarjeta de video nVidia Corporation Quadro FX 570, 4 cores y 32 GPU

LEAC CENTIC Supercomputacion Cluster Beowulf Cluster Integrador Heterogéneo

Cluster GIIB:

Nodos disponibles para el calculo: 3

Nodo1: nodo01-giib.uis.edu.co
Intel Pentium 4, 3.0 GHz , 512 MB Memoria RAM
interfaz de red Broadcom NetXtreme Gigabit Ethernet

Nodo2: nodo02-giib.uis.edu.co
Intel Pentium 4, 3.2 GHz , 512 MB Memoria RAM
interfaz de red Intel 82545GM Gigabit

Nodo3: nodo03-giib.uis.edu.co
Intel Pentium 4, 1.8 GHz , 256 MB de Memoria RAM
interfaz de red Intel 82540EM Gigabit Ethernet

Información sobre la plataforma:
http://sc3.uis.edu.co

Monitoreo de recursos:
http://frontend-leac.uis.edu.co/monika

Topología de la Red

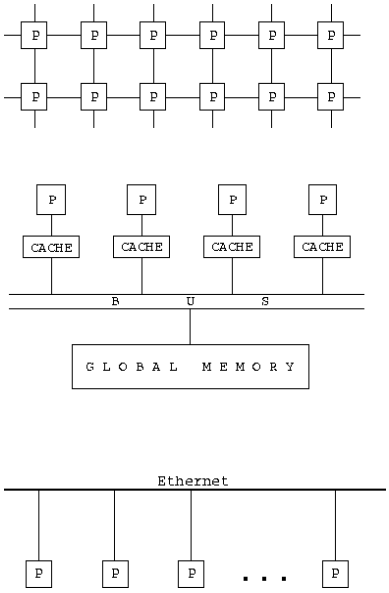
La topología por la cual se conectó la red entre los computadores fue de malla.

Modelo de Programación

El algoritmo propuesto se implementó en un modelo de programación SPMD (Single Program Multiple Data)

Arquitectura paralela

Figura 20. Arquitectura paralela del Cluster



Foster, I. 1995.

De arriba a abajo: una memoria de ordenador MIMD (Multiple Instruction Multiple Data) distribuido con una malla de interconexión, un multiprocesador de memoria compartida, y una red de área local (en este caso, una conexión Ethernet). En cada caso, P denota un procesador independiente.

3.5. Pruebas y Resultados

Para evaluar la versión paralela implementada respecto a la versión secuencial, se realizaron pruebas calculando el tiempo de procesamiento versus el número de nodos.

Para el análisis de los resultados que se obtuvieron en las pruebas, se utilizaron las métricas: Aceleración, Eficiencia, Ley de Amdahl, Ley de Gustafson y Comunicación en la red.

Capítulo 4

Diseño del Algoritmo Paralelo

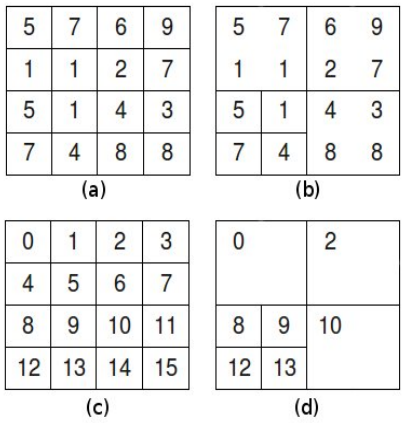
En este capítulo se presenta el diseño realizado de la versión paralela del algoritmo Split & Merge, empleando la metodología de Ian Foster.

4.1. Particionamiento

La fase Split es inherentemente paralela. Inicialmente, una imagen de tamaño $N = N_1 \times N_2$ es particionada en $P = P_1 \times P_2$ sub-imágenes que contienen píxeles adyacentes $N_1/P_1 \times N_2/P_2$. [27]La estructura para representar las regiones de la imagen es un árbol. [28]

[27] [MGG99]
[28] [MGG99], [TB90]

Figura 21. Fase Split: (a) valores de los píxeles (b) imagen particionada (c) números de identificación antes de la fase split (d) Ids regiones después de la primera y ultima iteración split



Autora Documento

4.2. Comunicación

Cada procesador trabaja dividiendo y fusionando regiones, lo que corresponde a ir eliminando o construyendo nodos en la estructura del árbol [29], las comunicaciones transfieren datos de los nodo adyacentes, esta información se almacena utilizando una matriz de adyacencia.

Cada procesador se comunica también con otros procesadores para mezclar regiones, satisfaciendo el criterio de homogeneidad (*Capitulo 3, sección 3.3*)
y las limitaciones (a) – (e) (*Capitulo 2, sección 2.2.3*)
en los pasos (1) – (3) (*Capitulo 2, sección 2.2.4*).

[29] [KG08]

Topología de grafo:

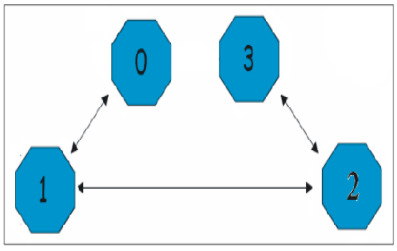
• Utilizado cuando la topología de los procesos es más genérica.

• Los nodos (procesos) tienen vecinos.

• La vecindad puede ser asimétrica (un nodo puede ser vecino de otro, pero no al revés).

Ejemplo:

Figura 22. Ejemplo Topología Grafo



Nodo	Vecinos	Índice	Aristas
0	1	1	1
1	2	3	0,2
2	2	5	1,3
3	1	6	2

Kruba Adali, Emrah Erkaan

4.3. Aglomeración

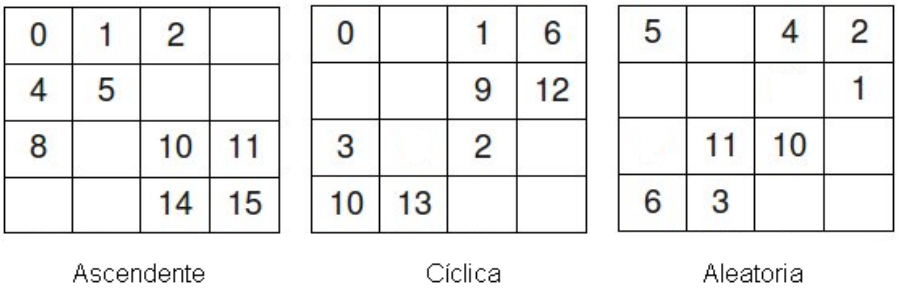
En la representación en quad-tree se define el método para la identificación de regiones (Id), para este método se utiliza una estrategia cíclica y aleatoria, cada procesador asigna un identificador a cada región [30]. Los identificadores deben ser diferentes para cada región y las regiones en cada procesador. El identificador asignado en el quad-tree contiene la información de los vértices y ejes de la región. Para seleccionar tareas concurrentemente se utiliza un balanceo de carga que se describe en el próximo paso.

Identificación de regiones:

- Ascendente
- Cíclica
- Aleatoria

Ejemplos:

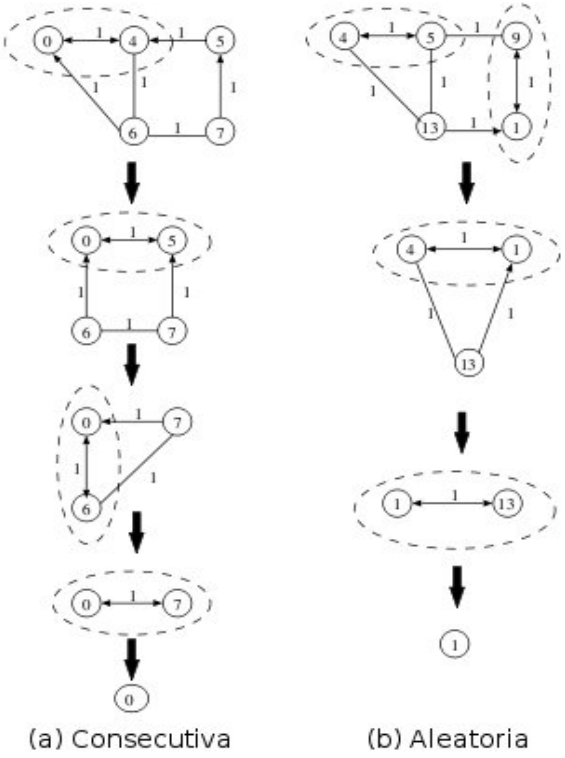
Figura 23. Ejemplos de métodos de identificación de regiones



Autora Documento

[30] [MGG03], [MGG99]

Figura 24. Ejemplo de la fase Merge



M. D. G. Montoya et al. 2003

4.4. Asignación

El proceso de fusión entre las regiones depende de los datos y el rendimiento de la comunicación, la aglomeración y la asignación del algoritmo disminuye con el tiempo de ejecución, en este caso los algoritmos de balance de carga estática y dinámica son necesarios. [31]

[31] [MGG03]

Algoritmo balance carga estática:

Antes de comenzar la fase Merge entre procesadores es conveniente reorganizar las tareas en los procesadores, ya que puede que un procesador tenga más carga que otro y tengan dependencia de datos.

Algoritmo balance carga dinámica:

Después de cada iteración de la fase Merge entre procesadores, es necesario reorganizar y redistribuir los procesos, ya que se pueden reducir las comunicaciones entre regiones que son homogéneas en sus límites.

4.5. Algoritmo Propuesto [32]

Fase Split:

Dado que es inherentemente paralela, no sufre modificaciones respecto a la versión secuencial del algoritmo Split & Merge.

Fase Merge:

Se modifica para producir fusiones de regiones en paralelo. Para el proceso de fusión es necesario que la secuencia de split sea ordenada.
El paradigma de fusión de regiones requiere que las regiones se mezclen únicamente con regiones vecinas que mejor satisfagan el criterio de homogeneidad, para ello todas las regiones chequean sus vecinos concurrentemente, siguiendo los pasos:

[32] [MGG03], [MGG99], [KG08], [TB90], [FPA+08]

1. Cada región, en un instante dado, solo se puede mezclar con otra única región, con la que mejor satisfaga el criterio de homogeneidad.
2. Para que dos regiones se mezclen, ambas deben haberse seleccionado mutuamente.
3. Un empate se rompe de forma aleatoria.

La fase *Merge se puede modelar usando un grafo no dirigido:*

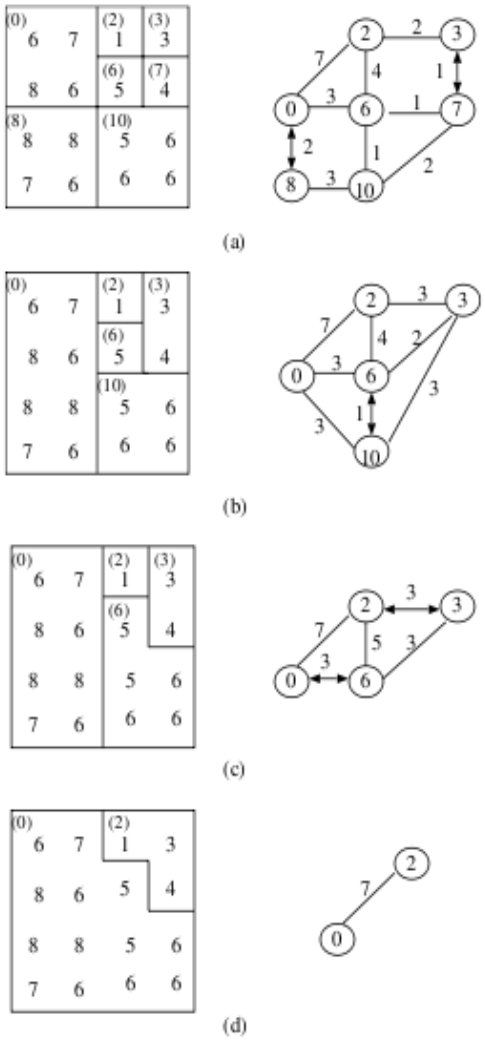
Suponiendo que G= (V,E) es un grafo no dirigido y sus ejes son representados por pesos. Los vértices del grafo V representan a las regiones en la imagen. Los ejes E están compuestos de los pares de vértices (v,w) tal que las regiones correspondientes a los vértices v y w comparten un límite en común.

El peso de un eje es igual al valor del criterio de homogeneidad evaluado para las regiones representadas por v y w y viene dado por:
ev.w = h(v,w) = H(u U v)

Para todos los ejes E, del grafo G, mezclar los vértices v y w, para los cuales ev,w es el eje de mínimo peso para ambos vértices v y w.
Para los vértices que poseen más de un eje con el mismo peso mínimo se selecciona el eje conectado con el vértice aleatoriamente. Se consideran sólo los ejes pesados que están dentro del umbral de homogeneidad.

Ejemplo:

Figura 25. Ejemplo Grafo Fase Merge: (a) inicia la fase merge (b) después de la primera iteración (c) después de la segunda iteración (d) después de la tercera iteración



M. D. G. Montoya et al. 2003

El algoritmo paralelo se describe en los siguientes pasos:

FASE SPLIT:

Paso I: Se particiona la imagen P1 x P2 en n sub-imágenes y se asignan sobre el conjunto de n procesadores.

Paso II: Cada procesador realiza el Split en su propia sub-imagen y asigna un identificador (ID) para cada una de las regiones creadas.

Paso III: Cada procesador construye su propio grafo local creando los conjunto de ejes entre las regiones locales que cumplan con el criterio de homogeneidad.

FASE MERGE:

Paso IV: Procesadores intercambian información sobre las regiones en la frontera de la sub-imagen inicial y se crea el conjunto de los ejes entre las regiones que pertenecen a diferentes vecinos del procesador.

Paso V: Ejecutar algoritmo de balanceo de carga estática.

Paso VI: Cada región determina las regiones seleccionadas que mejor se adapten a los criterios de homogeneidad para ser fusionadas. Si más de una región es candidata a fusionarse con ella en un empate, se seleccionará una de las regiones aleatoriamente.

Paso VII: Los procesadores intercambian información acerca de la mejor selección para las regiones de destino ubicados en diferentes procesadores.

Paso VIII: Los vértices y ejes del grafo se actualizan para el conjunto actual de las regiones.

Paso IX: Ejecutar algoritmo de balanceo de carga dinámica.

Paso X: Repita los pasos VI-*IX*, mientras que las regiones vinculadas todavía existen. De lo contrario el algoritmo termina.

Capítulo 5

Implementación del Algoritmo Paralelo

En este capítulo se definen los archivos de programación y funciones implementadas para la depuración del algoritmo, se explica el código principal del algoritmo, para el prototipo III como se expuso en el capítulo 3 y el diseño en el capítulo 4. Por último en este capítulo se presenta la Configuración del sistema para que pueda ejecutarse el algoritmo.

5.1. Descripción de Programas

Los programas y las funciones se explican en las siguientes tablas:

Tabla 1. Descripción de Programas

Programa	**Descripción**
Proyecto	Programa principal
Image	Contiene las funciones loadtiff, dumpTiff
Split	Contiene las funciones SplitImage, EtiquetarRegion, InfoRegion, CalculoNivelGris, CalculoCH, ComprobarCH

LocalMerge	Contiene las funciones MatrizAdyacencia, EntradaMerge, SeleccionarVecinos, MejorSeleccion, EmpateRegiones, MergeRegiones, ActualizarVertices, ActualizarEjes, ActualizarInfo, DesactivarEjes, ComprobarActivos
LimitesMerge	Contiene las funciones VerticesLimites, EjesLimites, SeleccionarVecinosLimites, MejorSeleccionLimite, MergeRegionesLimite, MergeRegionesLimite, ActualizarVerticesLimite, ActualizarEjesLimite, ActualizarInfoLimite, DesactivarEjesLimite, ComprobarActivosLimite
Grafo	Contiene las funciones CrearGrafo, CrearVertice, CrearEje, InfoRegion, DeleteVertice, DeleteEje, EtiquetarRegionAs, EtiquetarRegionCi, EtiquetarRegionRa
Balance	Contiene las funciones BalanceCargaEstatico, BalanceCargaDinamico

Tabla 2. Funciones del Programa

Funciones	**Descripción**
loadtiff	Función para Cargar la imagen
dumpTiff	Dump la imagen
SplitImage	Split rápido estrategia top-down
EtiquetarRegion	Etiqueta regiones creadas por el Split
InfoRegion	Almacena la información de la región creada por el Split
CalculoNivelGris	Calcula el nivel de gris de una región
CalculoCH	Calcula el criterio de homogeneidad de las regiones
ComprobarCH	Se comparan los criterios de homogeneidad calculados

	de las regiones en el split o merge
MatrizAdyacencia	Matriz de adyacencia entre regiones
EntradaMerge	Prepara el resultado del split para comenzar el merge
SeleccionarVecinos	Se seleccionan los vecinos a comprobar el CH para mezclar
MejorSeleccion	Compara las selecciones y elige la mejor
EmpateRegiones	En caso de empate entre regiones
MergeRegiones	Prepara para mezclar regiones a través de un grafo no dirigido
ActualizarVertices	Actualiza vértices del grafo local
ActualizarEjes	Actualiza ejes del grafo local
ActualizarInfo	Actualiza información de las regiones
DesactivarEjes	Desactivar ejes del grafo local
ComprobarActivos	Comprobar si hay ejes activos en el grafo para mezclar
VerticesLimites	Genera los vértices limite entre las regiones de diferentes procesadores
EjesLimites	Genera los ejes limite entre las regiones de diferentes procesadores
SeleccionarVecinosLimites	Se seleccionan entre vecinos de diferentes procesadores para mezclarse
MejorSeleccionLimite	Compara las selecciones de los limites y elige la mejor
MergeRegionesLimite	Prepara para mezclar regiones limite a través de un grafo no dirigido
ActualizarVerticesLimite	Actualiza vértices del grafo limite
ActualizarEjesLimite	Actualiza ejes del grafo limite
ActualizarInfoLimite	Actualiza la información de la región limite
DesactivarEjesLimite	Desactiva ejes de grafo limite
ComprobarActivosLimite	Comprobar si hay ejes activos para mezclar
CrearGrafo	Crear un grafo, inicio de la fase merge y merge entre regiones límites
CrearVertice	Crea un vértice del grafo

CrearEje	Crea un eje del grafo
InfoRegion	Almacena la información de la región
DeleteVertice	Borra un vértice del grafo
DeleteEje	Borra un eje del grafo
EtiquetarRegionAs	Etiqueta las regiones ascendentemente
EtiquetarRegionCi	Etiqueta las regiones cíclicamente
EtiquetarRegionRa	Etiqueta las regiones aleatoriamente
BalanceCargaEstatico	Algoritmo de balance de carga estático al inicio de la fase merge
BalanceCargaDinamico	Algoritmo de balance de carga dinámico en cada iteración de la fase merge

Para almacenar los datos durante la ejecución del algoritmo, se utilizaron básicamente las siguientes matrices:

Matriz	**Descripción**
Img[i][j]	Imagen Inicial
Ids[i][j]	Identificación de regiones
Info[i][j]	Información de las regiones
Ady[i][j]	Matriz de adyacencia
ImgEtiq[i][j]	Imagen Etiquetada

Para explicar mejor el algoritmo, se dividió el programa en varias partes, como se muestran a continuación.

Parte I : Inicialización de MPI; Definición del Sistema de Comunicaciones; Se carga la imagen; Se particiona la Imagen P1 x P2 en n sub-imágenes y se envía a los procesadores.

Parte II : Se realiza un Split Rápido estrategia Top Down en cada nodo con su propia sub-imagen utilizando la estrategia top down creando un quad-tree., sin calcular en criterio de homogeneidad, hasta un n/20. En la función se crea el árbol cuaternario, luego se comienza a calcular los nieles de gris y el CH de las regiones para continuar particionando. Durante el proceso se etiqueta la matriz Ids ascendentemente, se toma un píxel representativo de la región y la información se almacena en la matriz Info.

Parte III : Merge Local; Generación de los grafos locales; Identificación de regiones; Información de cada región.

Parte IV : Procesadores intercambian información sobre las regiones en la frontera de la sub-imagen inicial y se crea el conjunto de los ejes entre las regiones que pertenecen a diferentes vecinos del procesador.

Parte V : Algoritmo de balance de carga estático.
Antes de comenzar a fusionar regiones entre procesadores se equilibra la carga entre los procesadores, ya que puede que algunos nodos tengan muchas regiones que se puedan mezclar con otros procesadores y otros nodos tengan pocas regiones.

Parte VI : Merge entre regiones límite de cada nodo; Selección de regiones y comprobación de criterio de homogeneidad. Cada región determina las regiones seleccionadas que mejor se adapten a los criterios de homogeneidad para ser fusionadas. Si más de una región es candidata a fusionarse con ella en un empate, se seleccionará una de las regiones aleatoriamente.

Parte VII : Los procesadores intercambian información acerca de la mejor selección para las regiones de destino ubicados en diferentes procesadores.

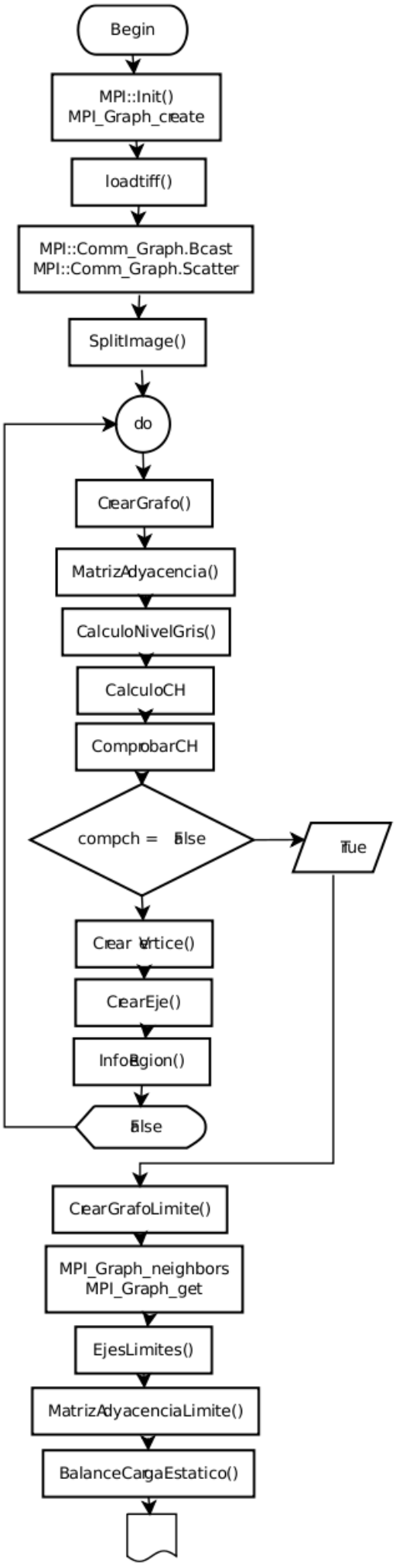
Parte VIII : Actualización de vértices y ejes.

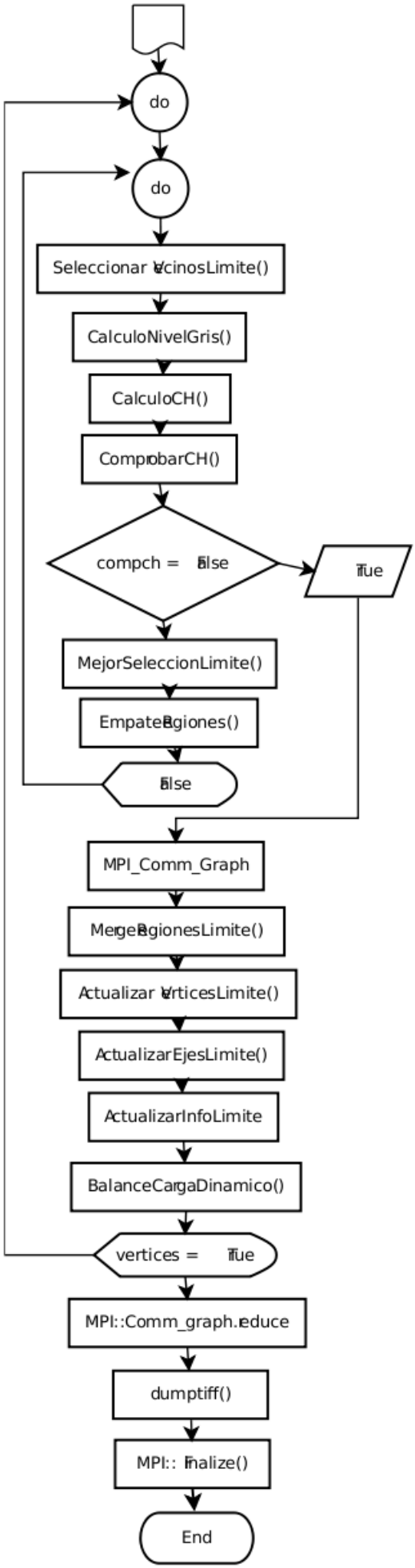
Parte IX : Algoritmo de balance de carga dinámico.
En cada iteración de la fase merge entre procesadores, se ejecuta este algoritmo para balancear la carga entre procesadores, si es que se requiere

Parte X : Salida: Array de regiones etiquetadas; Finalización del programa.
Repita mientras que las regiones vinculadas todavía existen. De lo contrario el algoritmo termina.

El algoritmo se representa por el siguiente diagrama de flujo:

Figura 26. Diagrama de Flujo del Algoritmo Implementado


Begin
MPI::Init()
MPI_Graph_create
loadtiff()
MPI::Comm_Graph.Bcast
MPI::Comm_Graph.Scatter
SplitImage()
do
CrearGrafo()
MatrizAdyacencia()
CalculoNivelGris()
CalculoCH
ComprobarCH
compch = false
true
CrearVertice()
CrearEje()
InfoRegion()
false
CrearGrafoLimite()
MPI_Graph_neighbors
MPI_Graph_get
EjesLimites()
MatrizAdyacenciaLimite()
BalanceCargaEstatico()


do
do
SeleccionarVecinosLimite()
CalculoNivelGris()
CalculoCH()
ComprobarCH()
compch = false
true
MejorSeleccionLimite()
EmpateRegiones()
false
MPI_Comm_Graph
MergeRegionesLimite()
ActualizarVerticesLimite()
ActualizarEjesLimite()
ActualizarInfoLimite
BalanceCargaDinamico()
vertices = true
MPI::Comm_graph.reduce
dumptiff()
MPI::Finalize()
End

5.2. Configuración del Sistema

El programa se implementó en un cluster, previamente descrito en el capítulo 3.4.5
Sistema operativo Linux
Lenguaje de programación gcc
Interfaz paso de mensajes Openmpi

Ejecución del programa principal:
mpiCC -o resultado Proyecto.cpp
mpirun -np 4 resultado

Resultados:
Tiempo de ejecución
Número de iteraciones

Capítulo 6

Pruebas de Ejecución

En este capítulo se evalúa el algoritmo luego de ejecutarse en el Cluster de computadoras, obteniendo los tiempos de ejecución de la versión secuencial y la versión paralela de acuerdo al número de procesadores, por último se grafican los resultados.

6.1. Evaluación del Rendimiento

De acuerdo a los tres prototipos diseñados y según las especificaciones de identificación de regiones, comunicaciones y algoritmos de balance de carga, se obtuvieron los siguientes resultados:

Prototipo I

Identificación: Ascendente

Comunicación: Global

Balance Carga: No

Tabla 3. Resultados Prototipo I, Id ascendente, Comm global, Sin Balance de carga

Tiempo (seg)	Número de Procesadores			
	1	2	3	4
Imagen 1	3210	3498	3115	2789
Imagen 2	2717	2914	2520	1980
Imagen 3	2420	2365	1918	1432
Imagen 4	3007	2894	2327	1696
Imagen 5	2502	2277	1632	1107
Imagen 6	2958	3105	2612	1983
Imagen 7	3088	2892	2477	1940
Imagen 8	3200	2899	1966	1236
Imagen 9	3309	3133	2745	2050

Prototipo II A

Identificación: Cíclico

Comunicación: Global

Balance Carga: Dinámico

Tabla 4. Resultados Prototipo II A, Id ascendente, Comm global, Balance de carga dinámico

Tiempo (seg)	Número de Procesadores			
	1	2	3	4
Imagen 1	3150	3042	2845	2114
Imagen 2	2677	2435	1978	1523
Imagen 3	2355	2111	1788	1545
Imagen 4	2447	2006	1734	1392
Imagen 5	2095	1711	1203	943
Imagen 6	2835	2921	2744	2132
Imagen 7	2609	2720	2160	1533
Imagen 8	2805	2380	1609	1128
Imagen 9	2987	2589	1992	1399

Prototipo II B

Identificación: Aleatoria

Comunicación: Global

Balance Carga: Dinámico

Tabla 5. Resultados Prototipo II B, Id cíclico, Comm global, Balance de carga dinámico

Tiempo (seg)	Número de Procesadores			
	1	2	3	4
Imagen 1	2984	2232	1940	1587
Imagen 2	2280	1941	1590	1409
Imagen 3	2158	1835	1247	921
Imagen 4	2109	1882	1520	1021
Imagen 5	1833	1021	688	489
Imagen 6	2551	2107	1630	1288
Imagen 7	2376	1842	1300	909
Imagen 8	2054	1614	1080	514
Imagen 9	2607	2022	1751	1325

Prototipo III A

Identificación: Cíclico

Comunicación: Topología Grafo

Balance Carga: Estático y Dinámico

Tabla 6. Resultados Prototipo III A, Id cíclico, Comm topo-grafo, Balance de carga estático y dinámico

Tiempo (seg)	Número de Procesadores			
	1	2	3	4
Imagen 1	2991	2121	1376	978
Imagen 2	2133	2003	1689	1211
Imagen 3	2299	1791	1103	889
Imagen 4	2474	1925	1309	790
Imagen 5	1745	994	763	518
Imagen 6	2670	2056	1505	1102
Imagen 7	2489	1930	1507	1078
Imagen 8	1890	1301	978	520
Imagen 9	2762	2184	1629	1131

Prototipo III B

Identificación: Aleatoria

Comunicación: Topología Grafo

Balance Carga: Estático y Dinámico

Tabla 7. Resultados Prototipo III B, Id aleatoria, Comm topo-grafo, Balance de carga estático y dinámico

Tiempo (seg)	Número de Procesadores			
	1	2	3	4
Imagen 1	2896	1944	1233	812
Imagen 2	1965	1663	945	612
Imagen 3	2006	1821	973	559
Imagen 4	1966	1223	762	345
Imagen 5	1520	1118	754	432
Imagen 6	2490	2180	1483	915
Imagen 7	2200	1596	948	642
Imagen 8	1504	950	605	244
Imagen 9	2480	1944	1233	1021

6.2. Graficación de Resultados

Imagen 1:

Gráfica 1. Resultados Tiempo de Procesamiento Imagen 1

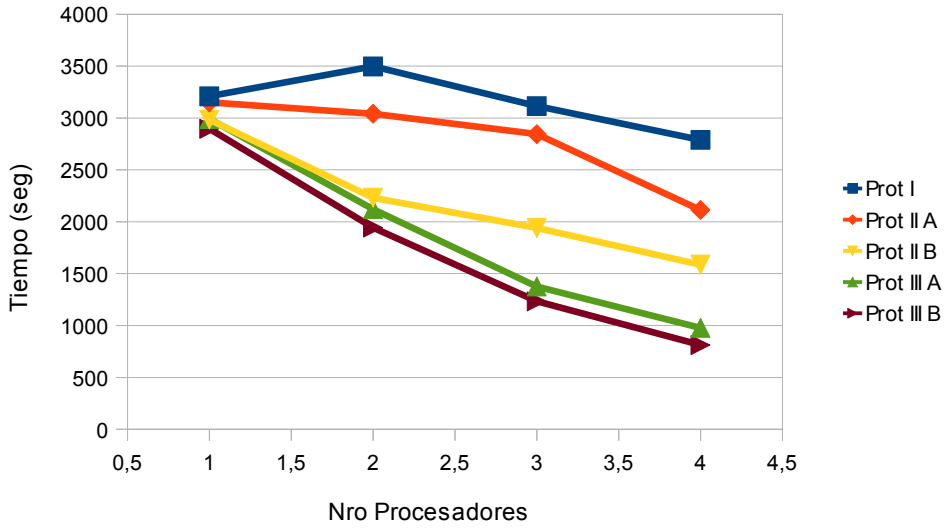


Imagen 2:

Gráfica 2. Resultados Tiempo de Procesamiento Imagen 2

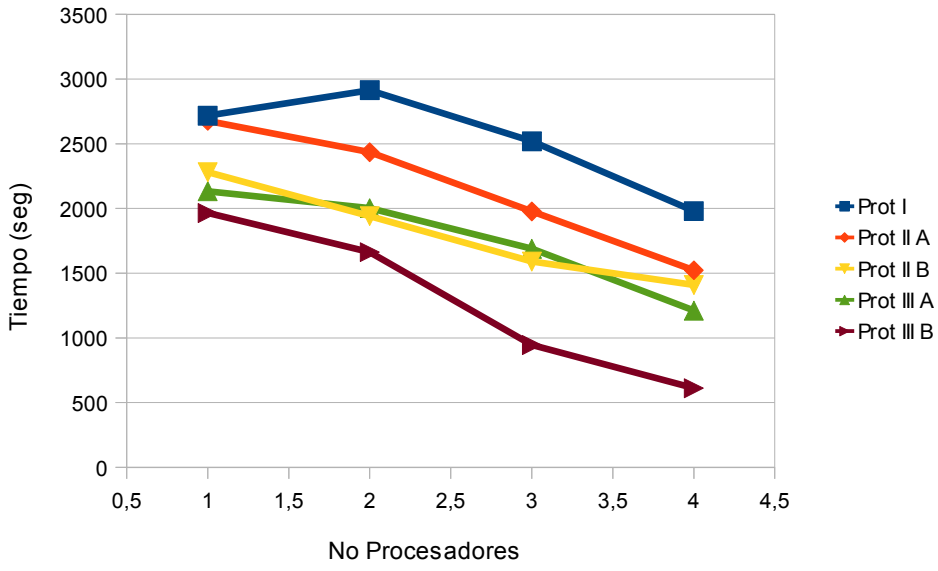


Imagen 3:

Gráfica 3. Resultados Tiempo de Procesamiento Imagen 3

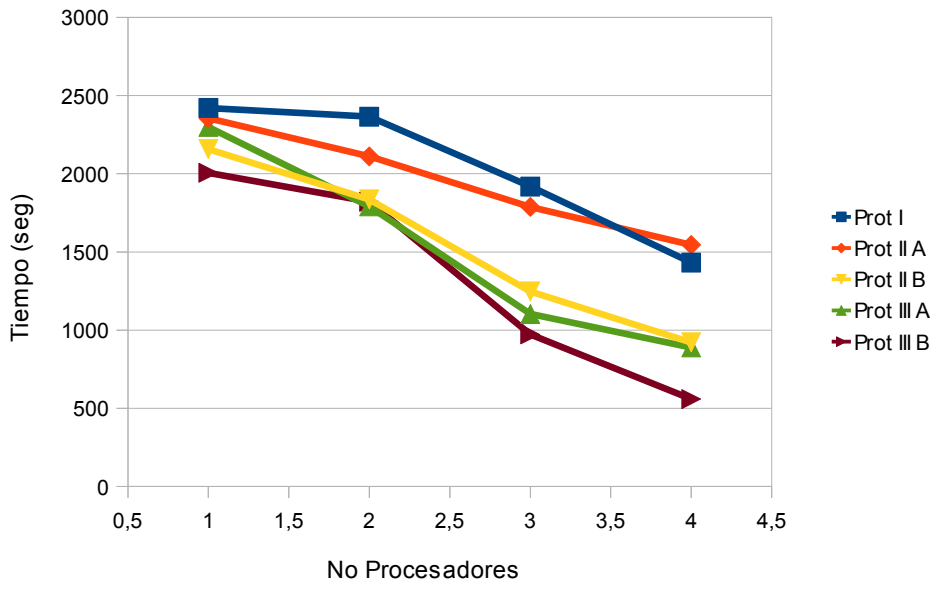


Imagen 4:

Gráfica 4. Resultados Tiempo de Procesamiento Imagen 4

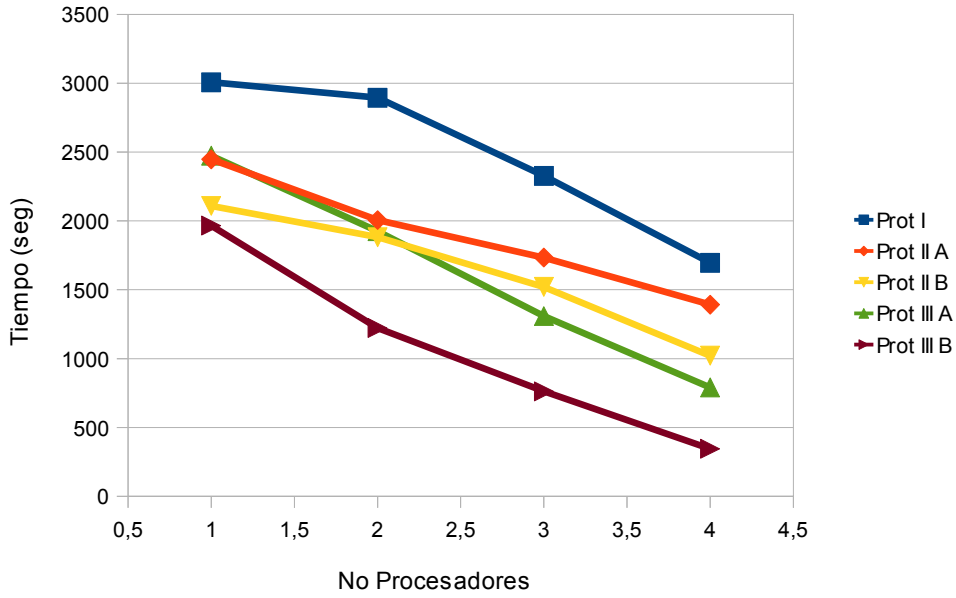


Imagen 5:

Gráfica 5. Resultados Tiempo de Procesamiento Imagen 5

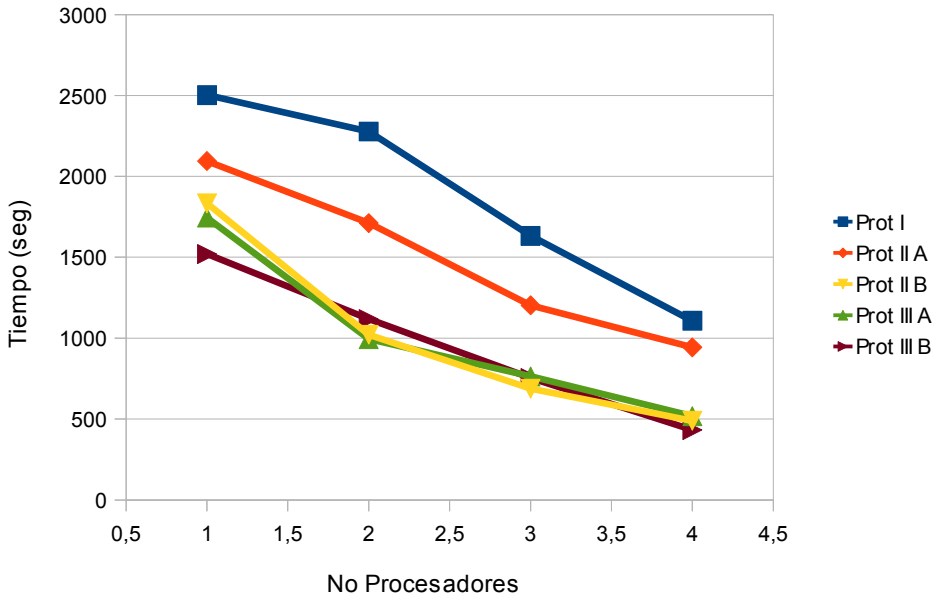


Imagen 6:

Gráfica 6. Resultados Tiempo de Procesamiento Imagen 6

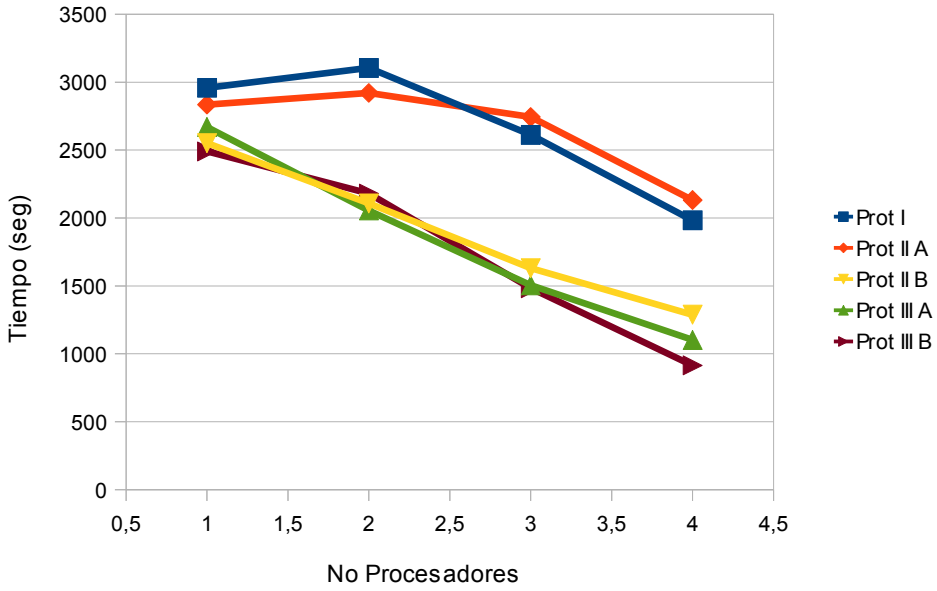


Imagen 7:

Gráfica 7. Resultados Tiempo de Procesamiento Imagen 7

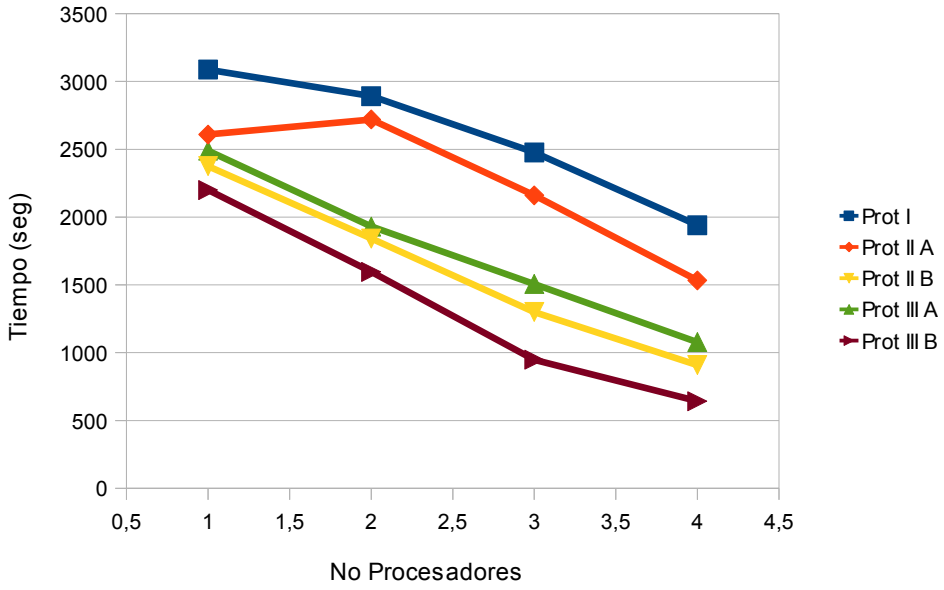


Imagen 8:

Gráfica 8. Resultados Tiempo de Procesamiento Imagen 8

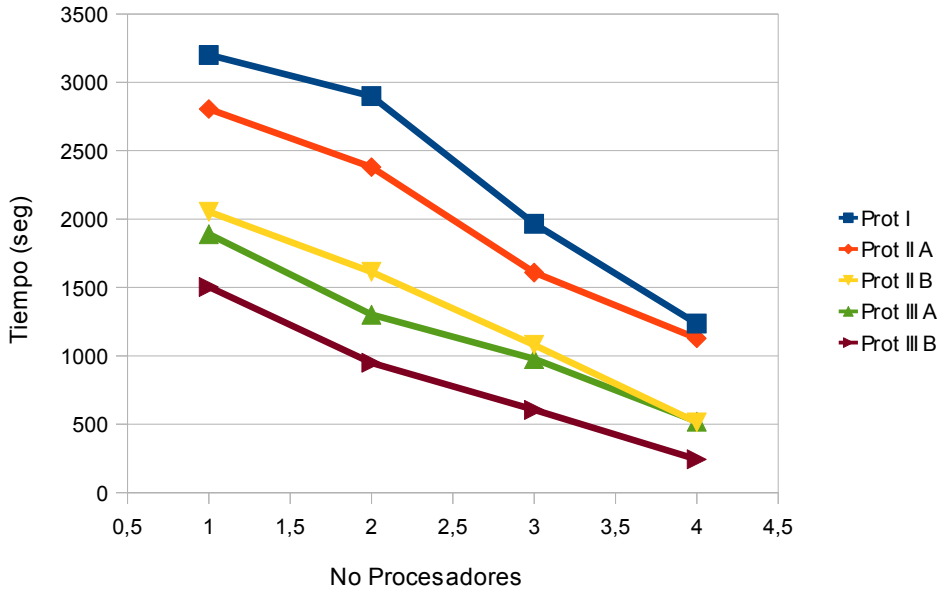
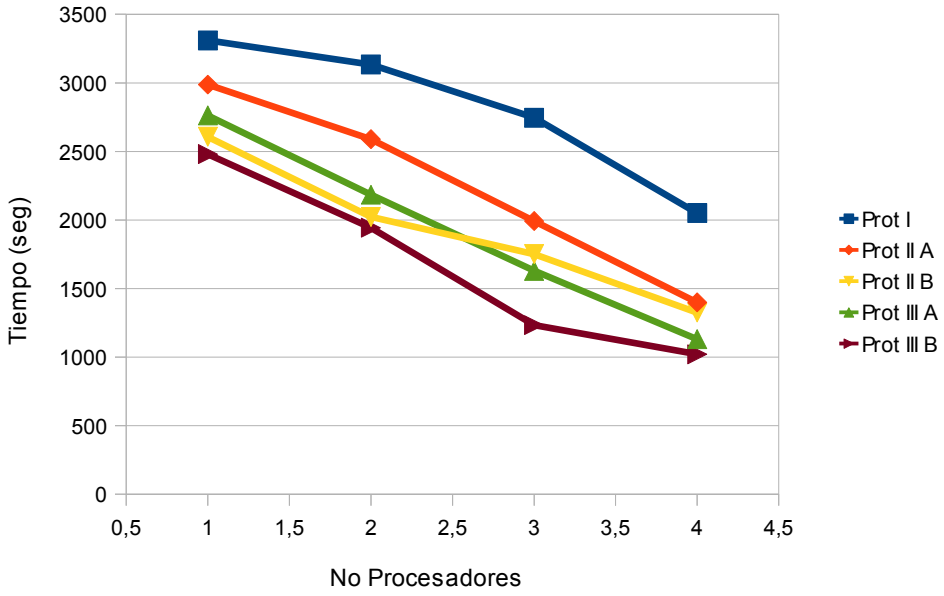


Imagen 9:

Gráfica 9. Resultados Tiempo de Procesamiento Imagen 9



Se puede observar que los mejores resultados en tiempos de ejecución los han obtenido los prototipos II y III, en los cuales se introdujo un sistema de balanceo de carga, y cambio en las comunicaciones en el caso del prototipo III, para el cual se obtuvo el mejor resultado con la versión B.

En caso contrario el prototipo I obtiene el peor resultado, poniendo en manifiesto la importancia de la forma de identificar las regiones, en este caso la forma ascendente no es la más indicada porque crea una secuencialidad de las mezclas lo que genera un gran desbalance de carga y al no ejecutarse ningún algoritmo de balanceo el tiempo de procesamiento es más elevado.

La forma cíclica de identificar las regiones ayuda a disminuir los desbalanceos, pero la forma de identificar aleatoriamente es la que más ayuda a reducirlos. Lo

cual también se puede apreciar observando que el número de iteraciones disminuye entre los prototipos con Id ascendente, cíclica y aleatoria.

Otro factor que ayudó al rendimiento de algoritmo, fueron las comunicaciones, se hace notable entre los prototipos II y III.

De acuerdo a lo anterior se puede concluir que los algoritmos de balance de carga estático y dinámico ayudan a mejorar el rendimiento, de igual forma la optimización en las comunicaciones.

Capítulo 7

Análisis de Resultados

En este capítulo se analizan los resultados obtenidos en las pruebas de ejecución, realizando la comparación entre la versión secuencial y la paralela del algoritmo, determinadas por las métricas de Aceleración, ley de Amdahl, ley de Gustafson, Eficiencia y la capacidad de comunicación en la red.

7.1. Aceleración y Eficiencia

La aceleración está definida como:
$S(n) = T(1) / T(n)$

y la eficiencia está definida como:
$E(n) = T(1) / nT(n)$

donde T(1) es el tiempo de ejecución en 1 procesador y T(n) es el tiempo de ejecución en n procesadores

La eficiencia es una comparación del grado de aceleración conseguido frente al valor máximo.
Dado que $1 \leq S(n) \leq n$, tenemos $1/n \leq E(n) \leq 1$.

La eficiencia máxima (E(n) = 1) se obtiene cuando todos los procesadores están siendo completamente utilizados durante todo el periodo de ejecución.

La ley de Amdahl se usa para averiguar la mejora máxima de un sistema cuando solo una parte de éste es mejorado. Pone límite a la mejora por Aceleración que puede tener debida a la paralelización. Ofreciendo así una visión pesimista del procesamiento paralelo.

La ley de Amdahl está definida como:
LA(n) = n / (1+(n-1)*s) , donde s es el porcentaje de código no paralelizable

La eficiencia máxima está dada como:
E(n)max= 1 / 1+(n-1)*s

La ley de Gustafson establece que cualquier problema suficientemente grande puede ser eficientemente paralelizado. La ley de Gustafson esta muy ligada a Ley de Amdahl que por el contrario la ley de Gustafson ofrece un nuevo punto de vista y así una visión positiva de las ventajas del procesamiento paralelo.

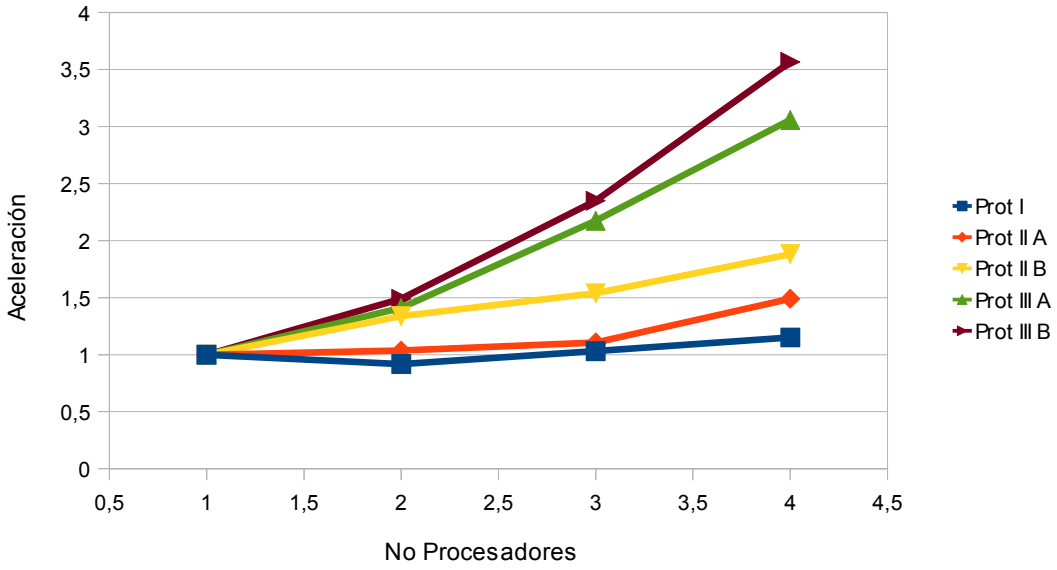
La ley de Gustafson está definida como:
LG(n) = n – s*(n-1) , donde s es el porcentaje de código no paralelizable

Para evaluar los indicadores se eligió el prototipo III como prototipo final ya que obtuvo los tiempos de ejecución más bajos que los demás prototipos. El valor de s se estimó aproximadamente un 15% de código no paralelizable.

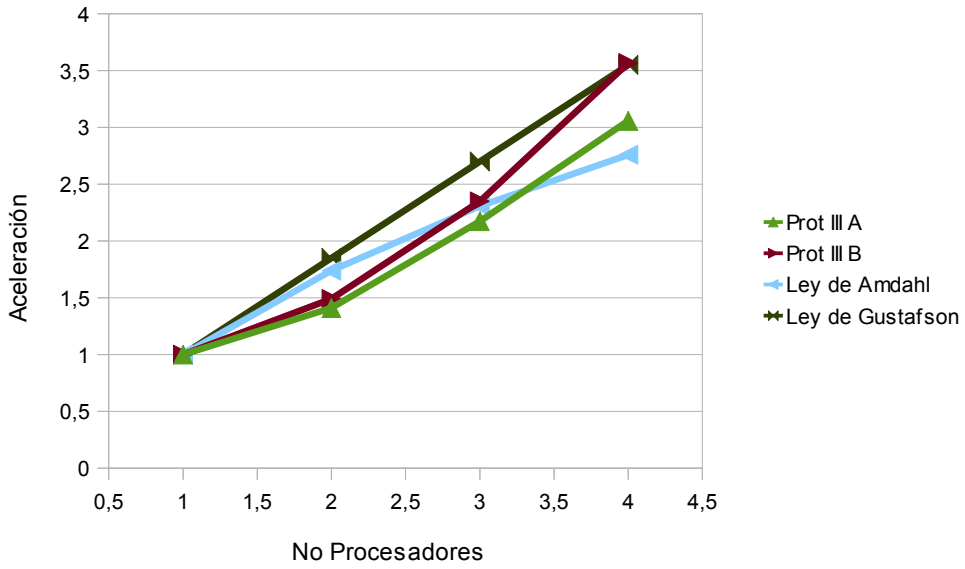
A continuación se muestran los cálculos obtenidos por las métricas:

Imagen 1:

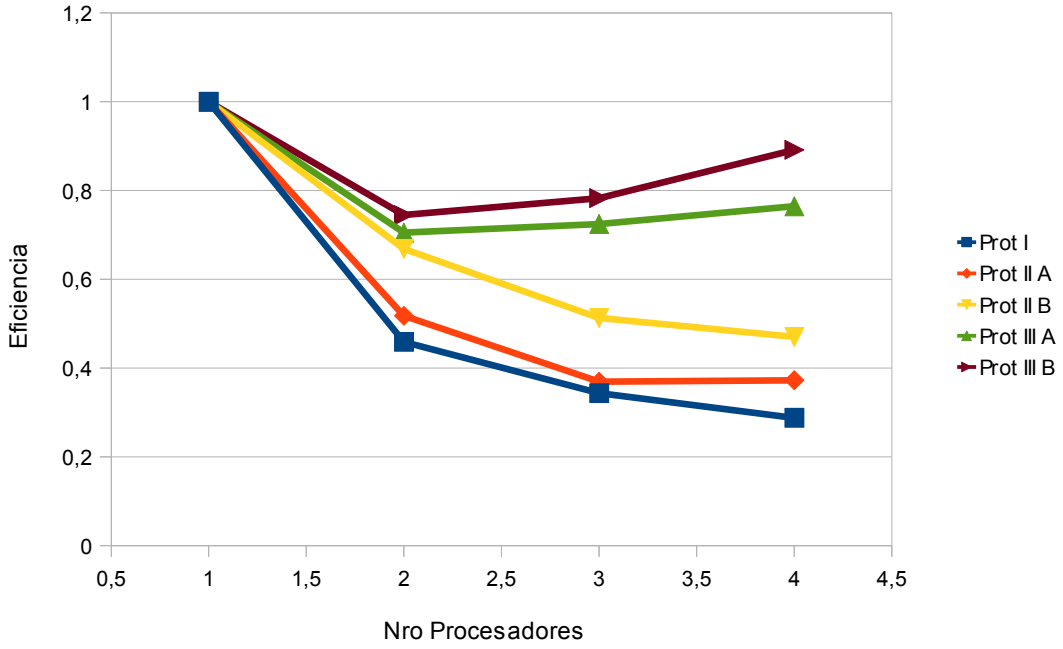
Gráfica 10. Resultados Aceleración Imagen 1



Gráfica 11. Resultados Aceleración, Ley de Amdahl, Ley de Gustafson, Imagen 1



Gráfica 12. Resultados Eficiencia Imagen 1



Gráfica 13. Resultados Eficiencia Teórica Imagen 1

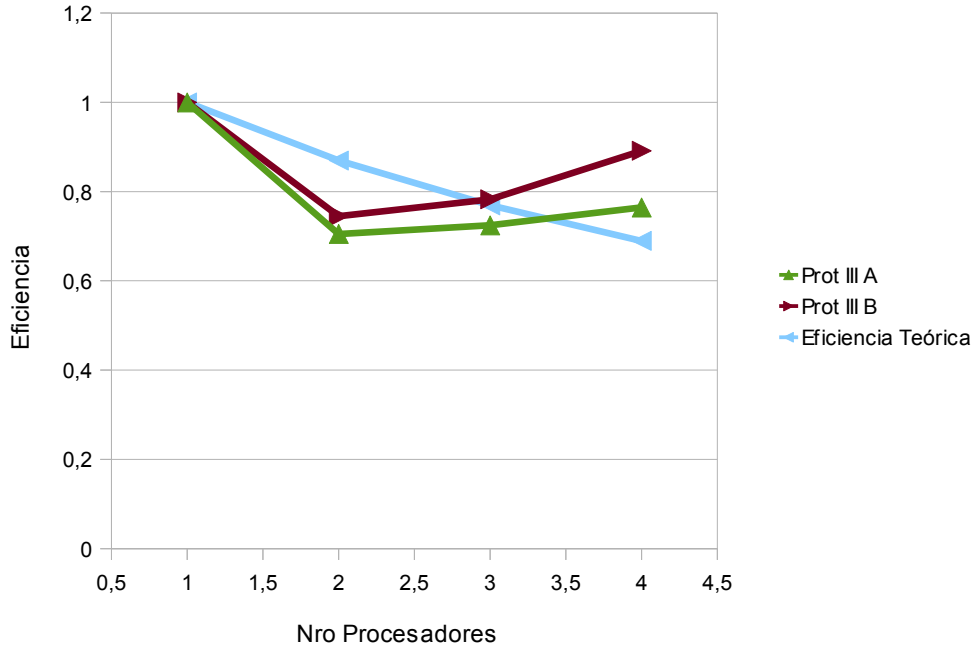
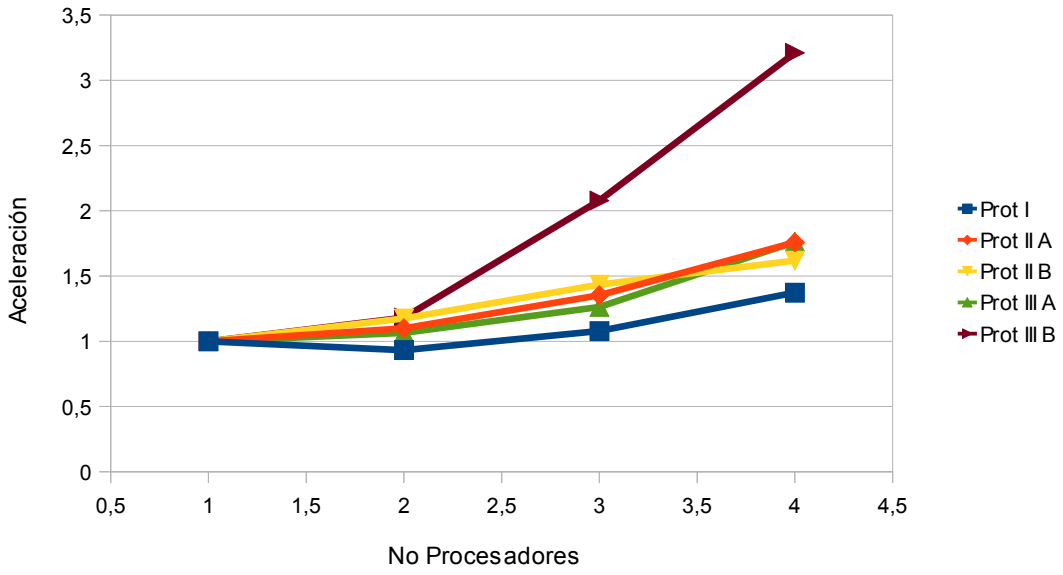
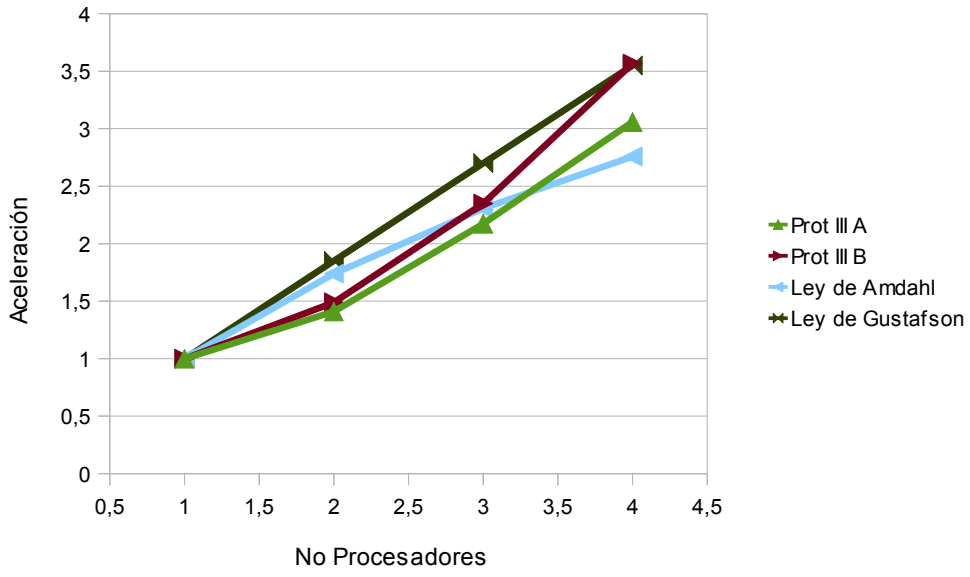


Imagen 2:

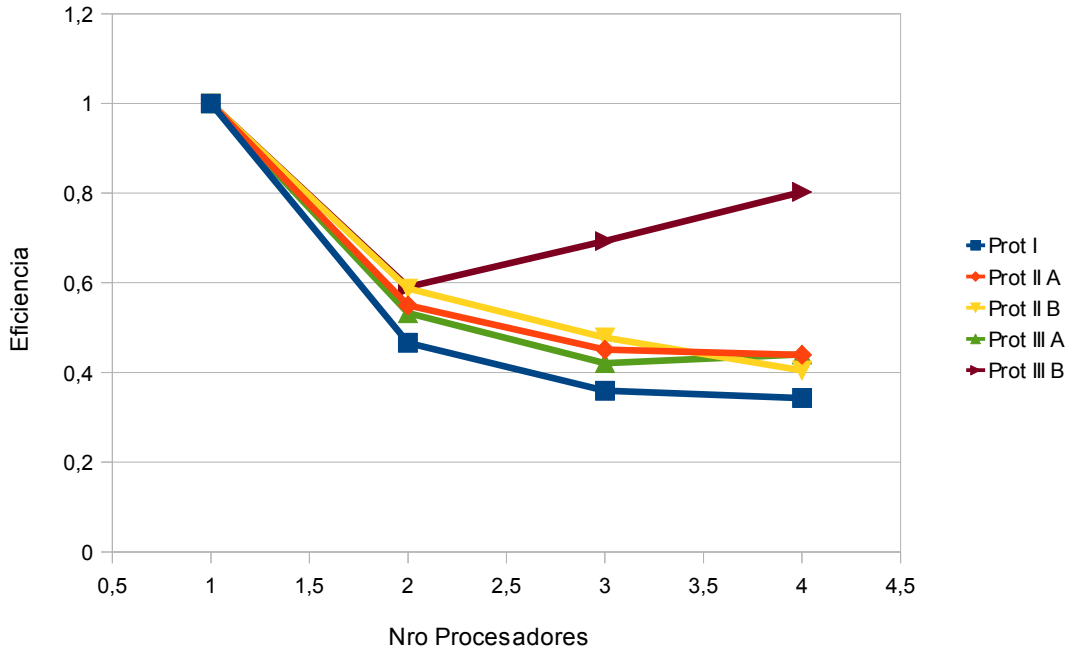
Gráfica 14. Resultados Aceleración Imagen 2



Gráfica 15. Resultados Aceleración, Ley de Amdahl, Ley de Gustafson, Imagen 2



Gráfica 16. Resultados Eficiencia Imagen 2



Gráfica 17. Resultados Eficiencia Teórica Imagen 2

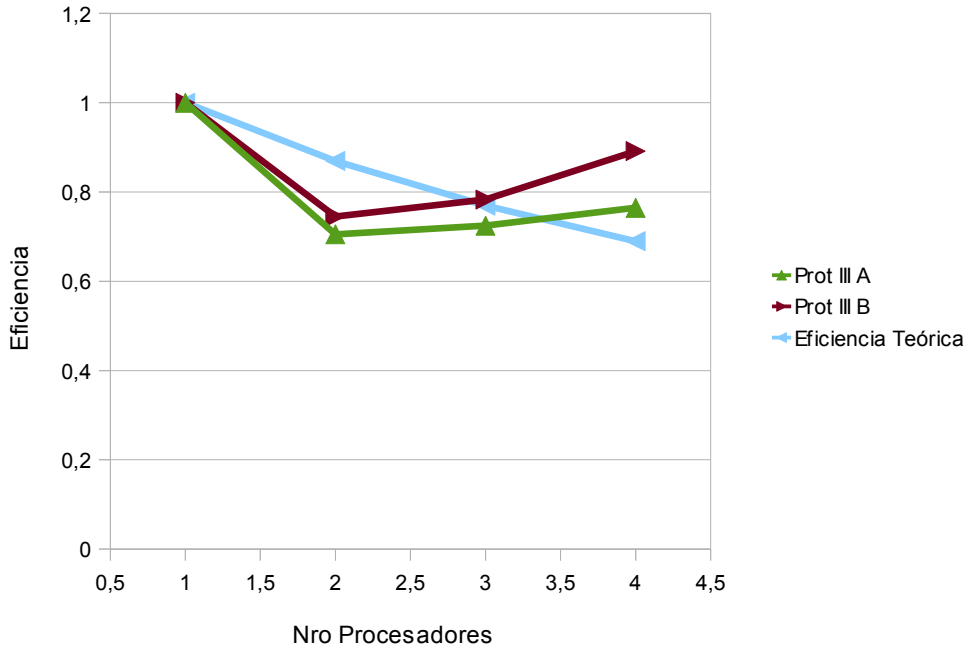
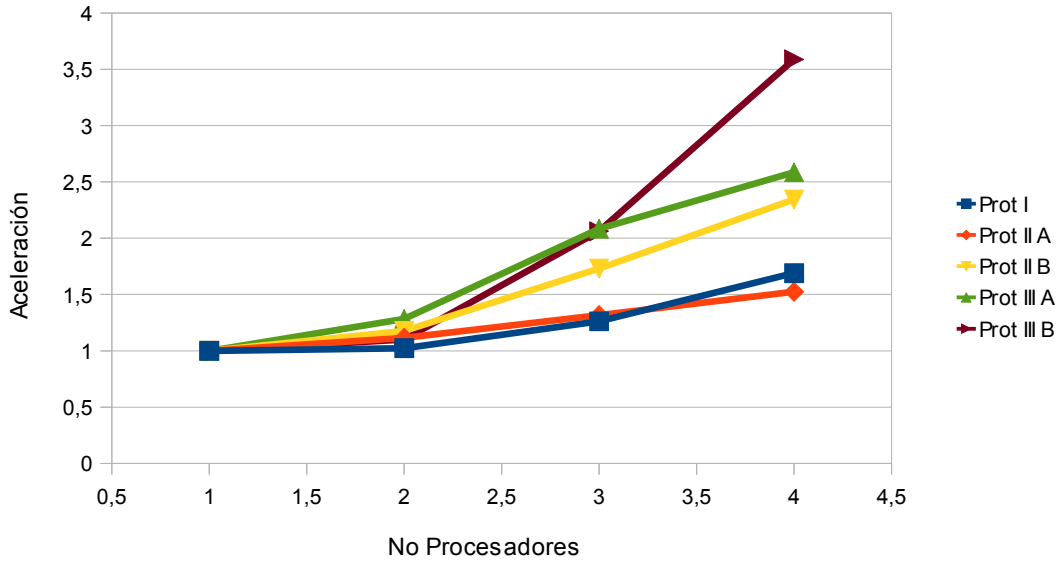
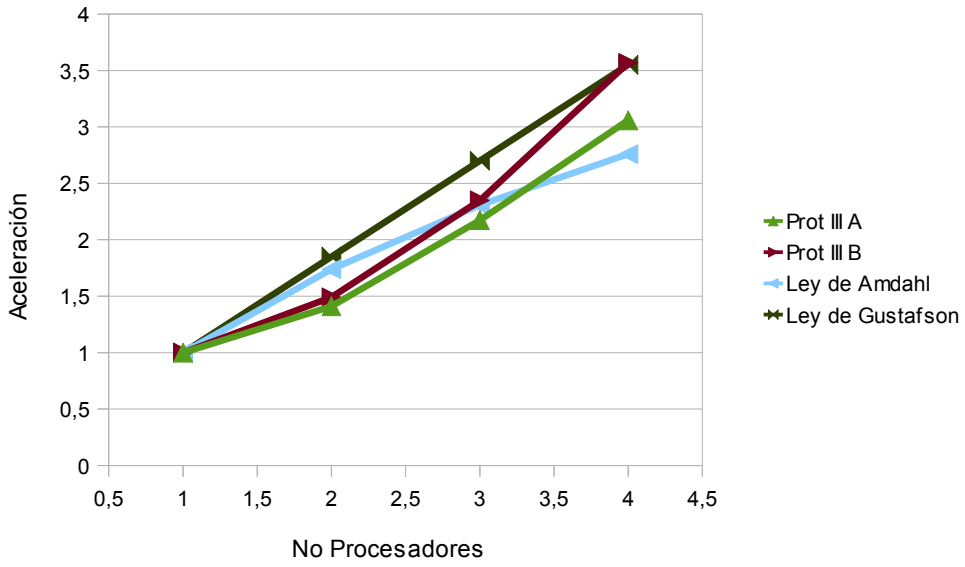


Imagen 3:

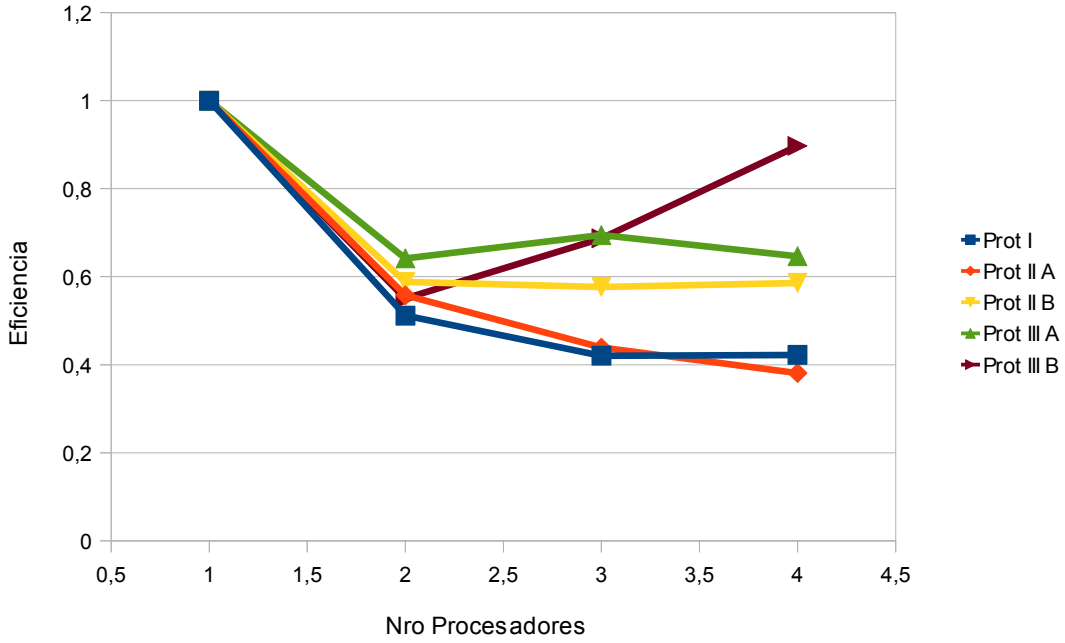
Gráfica 18. Resultados Aceleración Imagen 3



Gráfica 19. Resultados Aceleración, Ley de Amdahl, Ley de Gustafson, Imagen 3



Gráfica 20. Resultados Eficiencia Imagen 3



Gráfica 21. Resultados Eficiencia Teórica Imagen 3

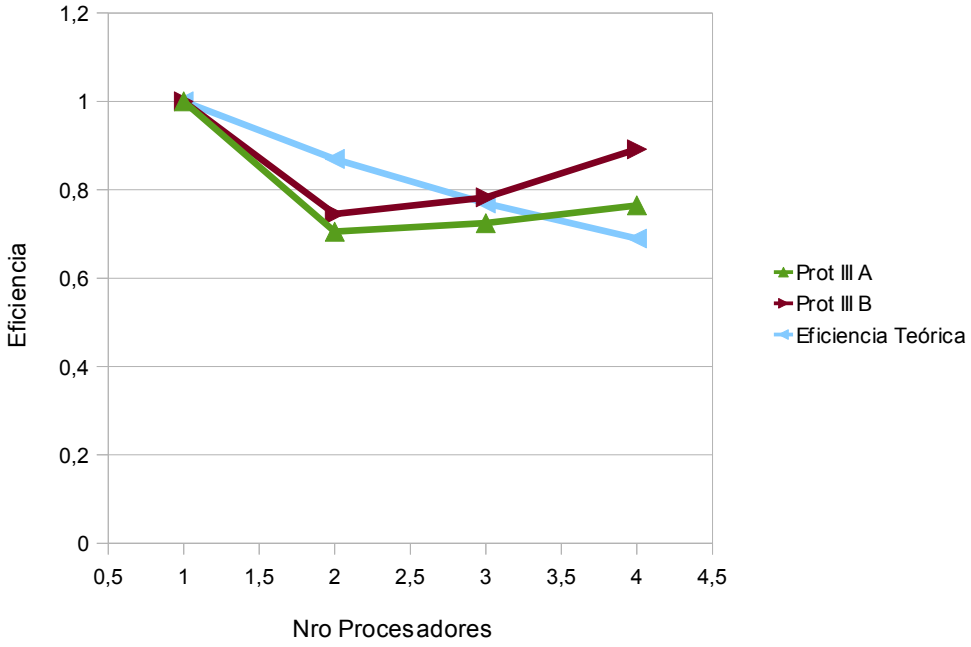
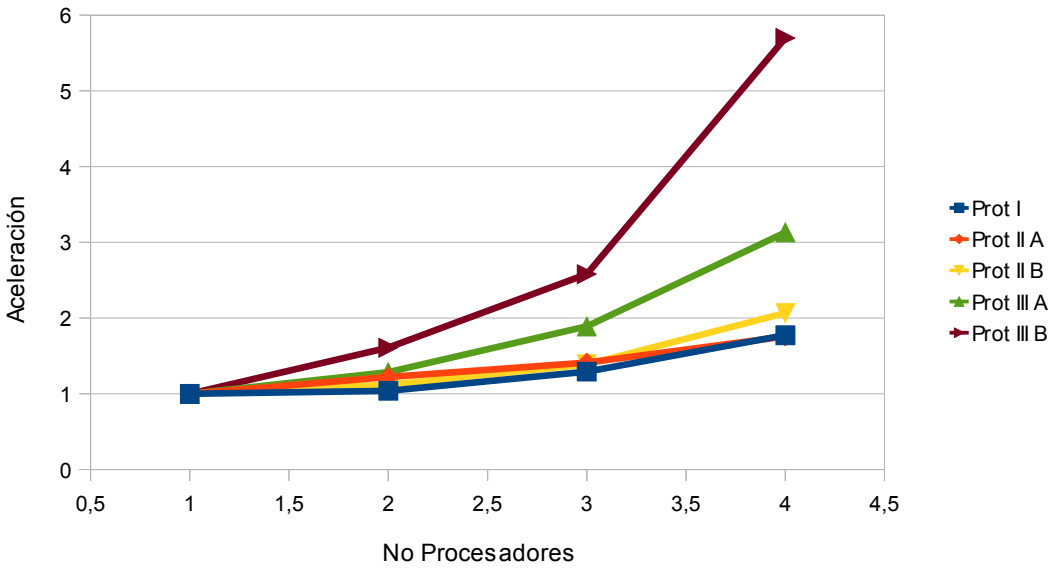
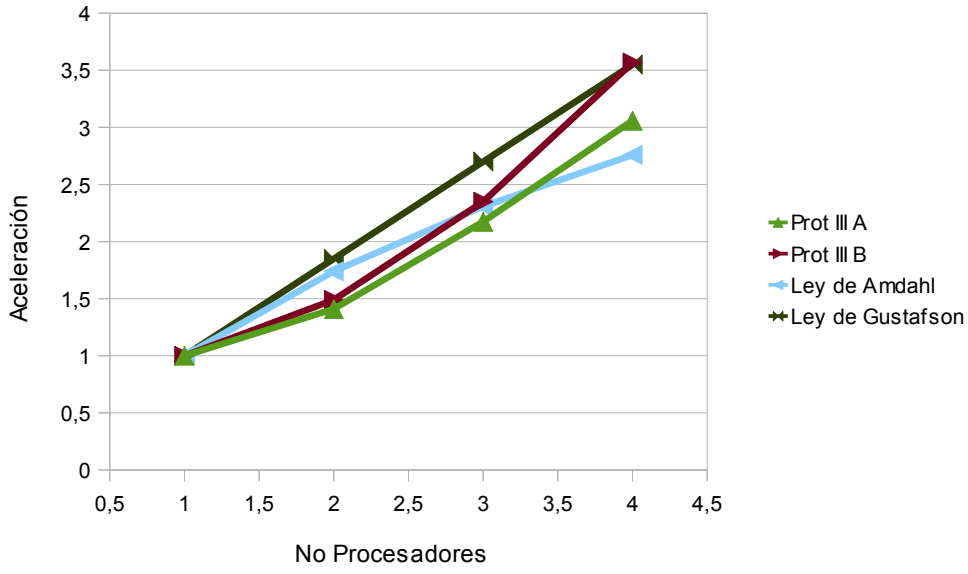


Imagen 4:

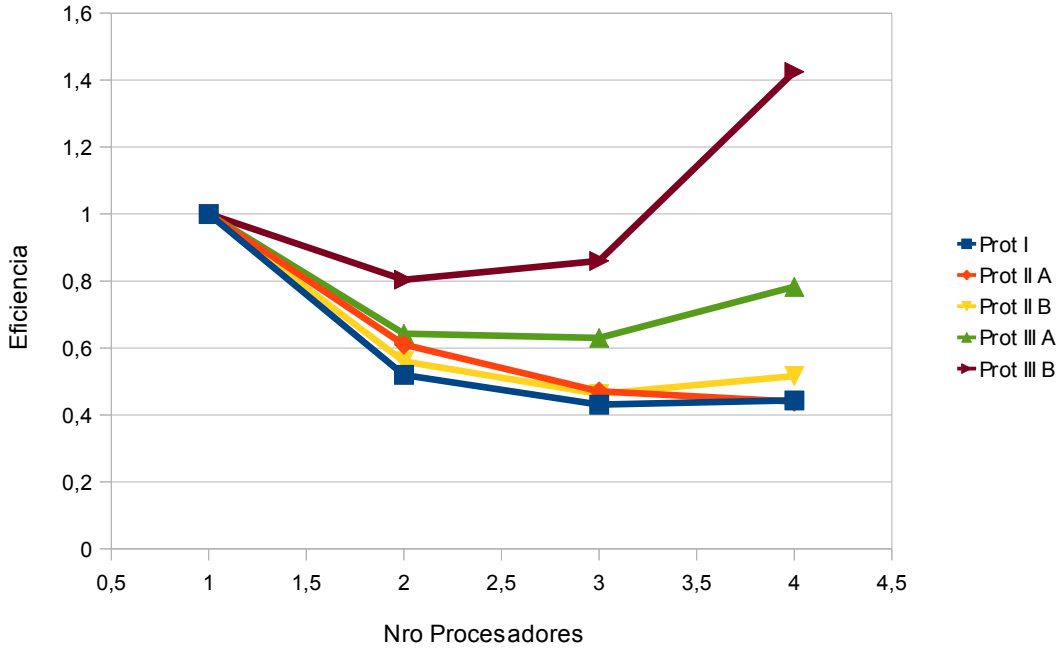
Gráfica 22. Resultados Aceleración Imagen 4



Gráfica 23. Resultados Aceleración, Ley de Amdahl, Ley de Gustafson, Imagen 4



Gráfica 24. Resultados Eficiencia Imagen 4



Gráfica 25. Resultados Eficiencia Teórica Imagen 4

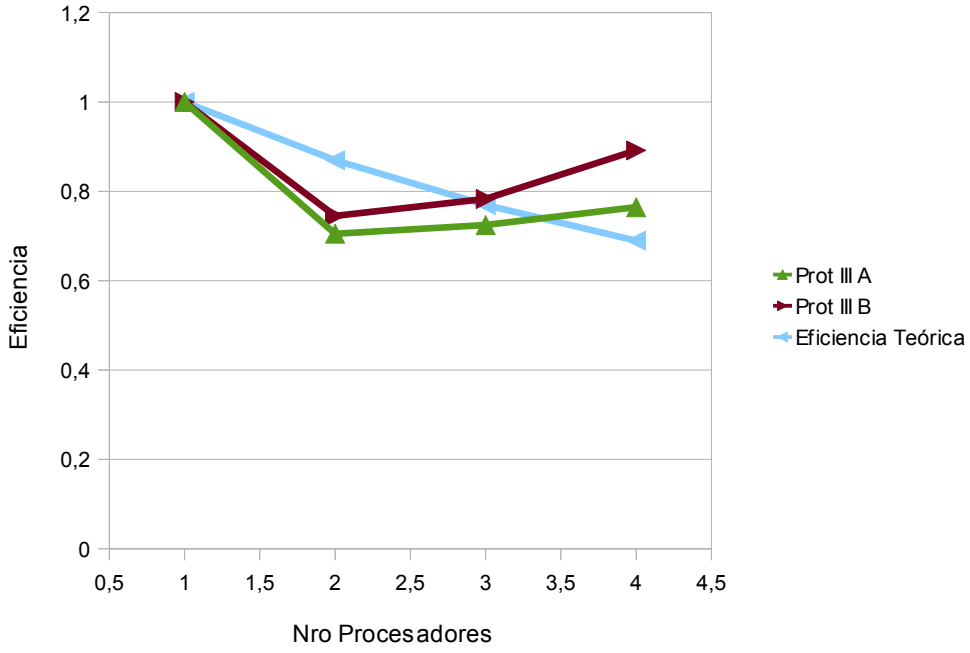
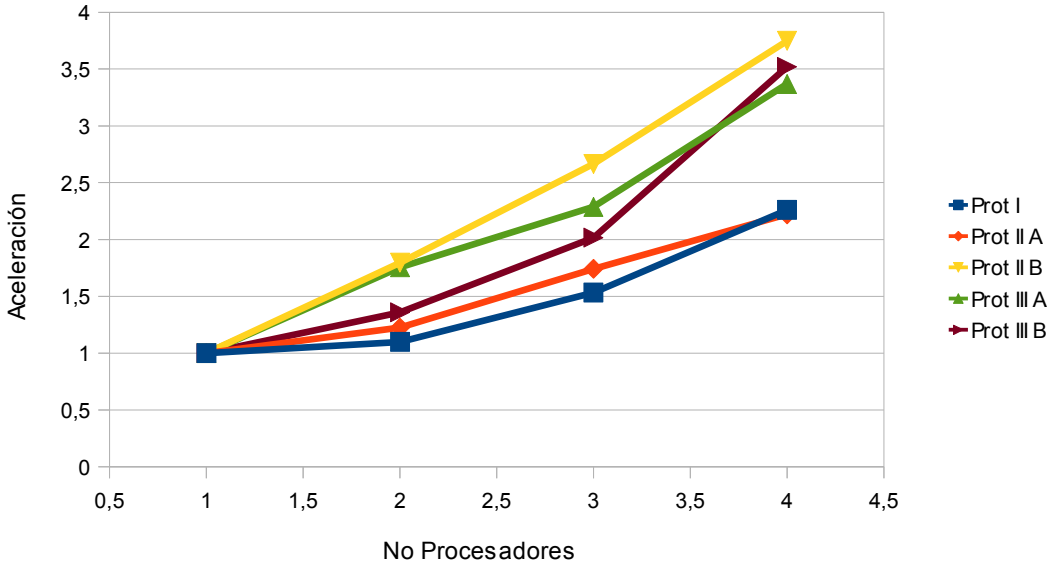
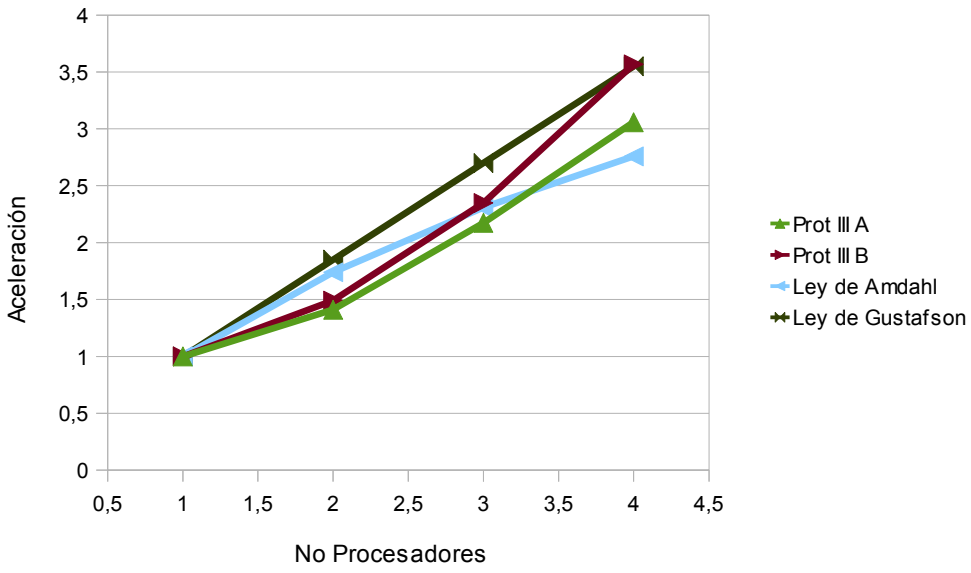


Imagen 5:

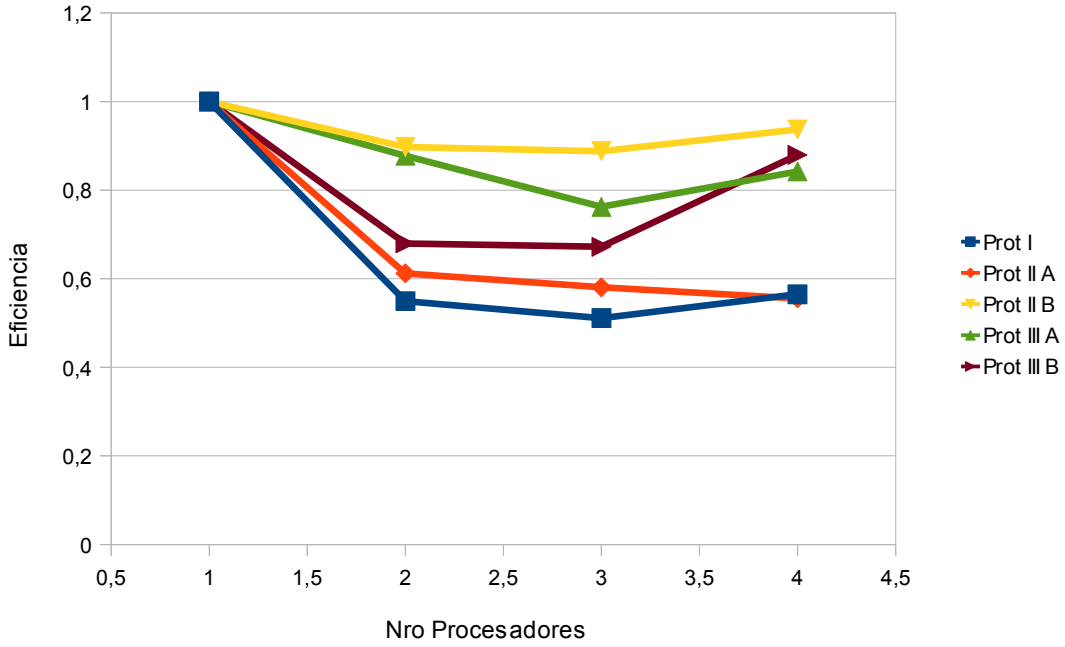
Gráfica 26. Resultados Aceleración Imagen 5



Gráfica 27. Resultados Aceleración, Ley de Amdahl, Ley de Gustafson, Imagen 5



Gráfica 28. Resultados Eficiencia Imagen 5



Gráfica 29. Resultados Eficiencia Teórica Imagen 5

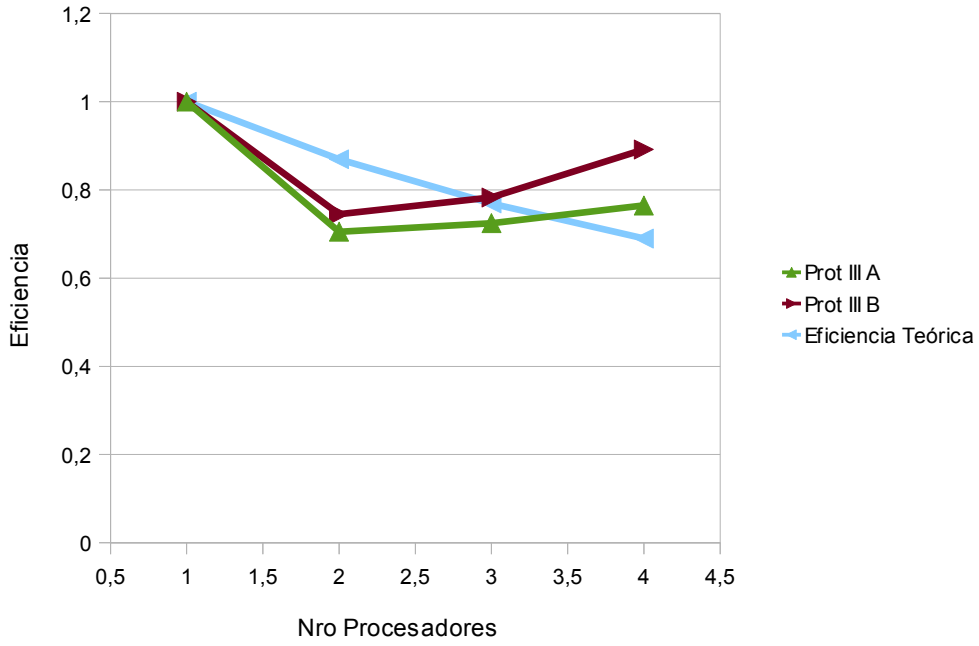
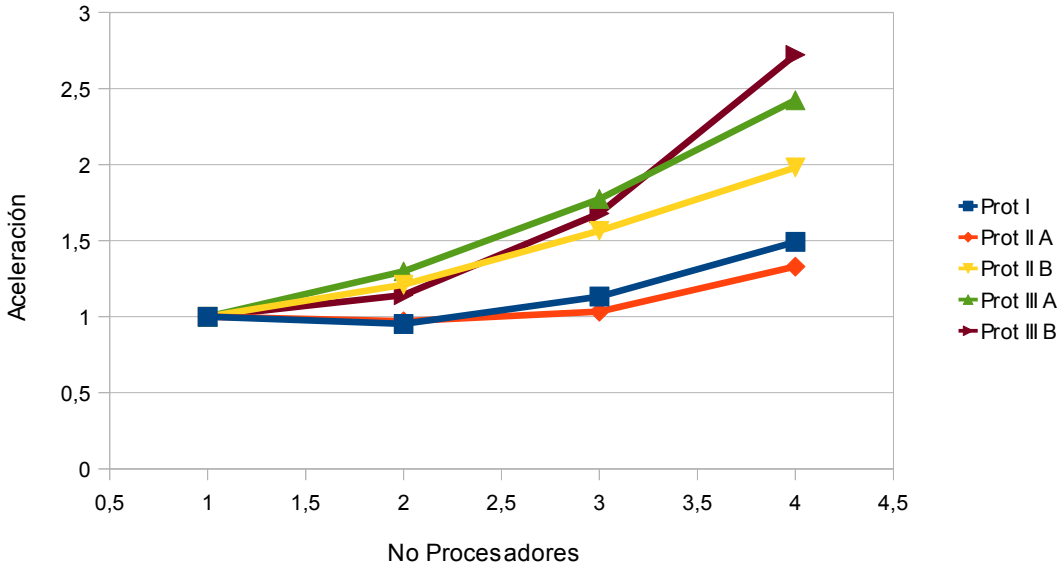
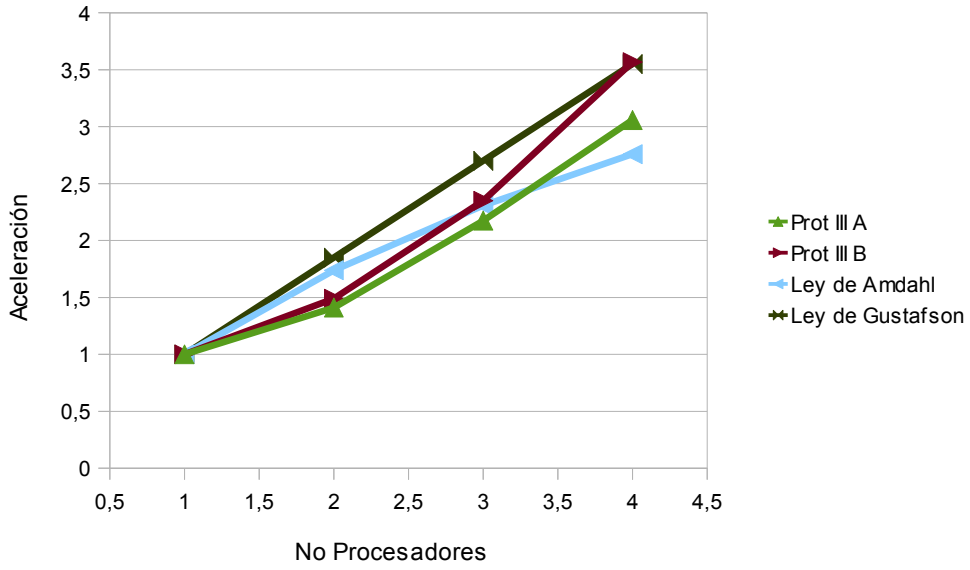


Imagen 6:

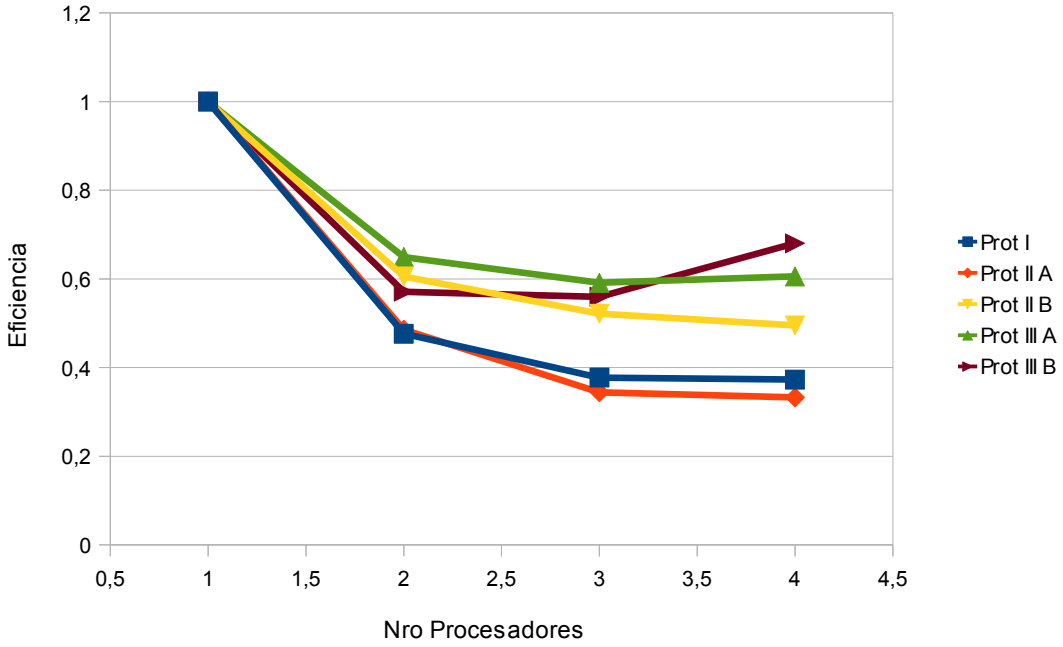
Gráfica 30. Resultados Aceleración Imagen 6



Gráfica 31. Resultados Aceleración, Ley de Amdahl, Ley de Gustafson, Imagen 6



Gráfica 32. Resultados Eficiencia Imagen 6



Gráfica 33. Resultados Eficiencia Teórica Imagen 6

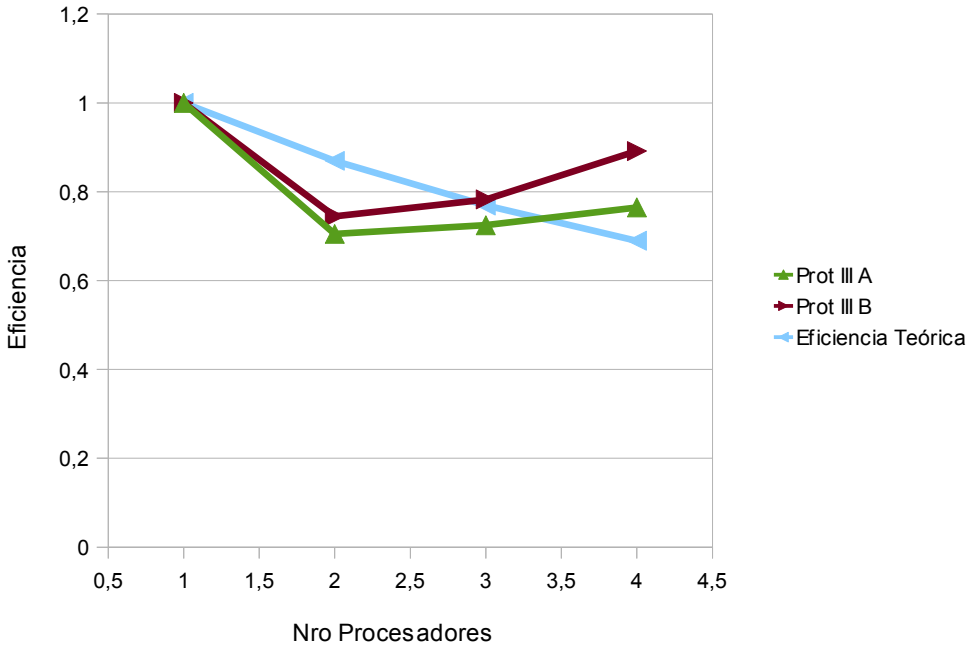
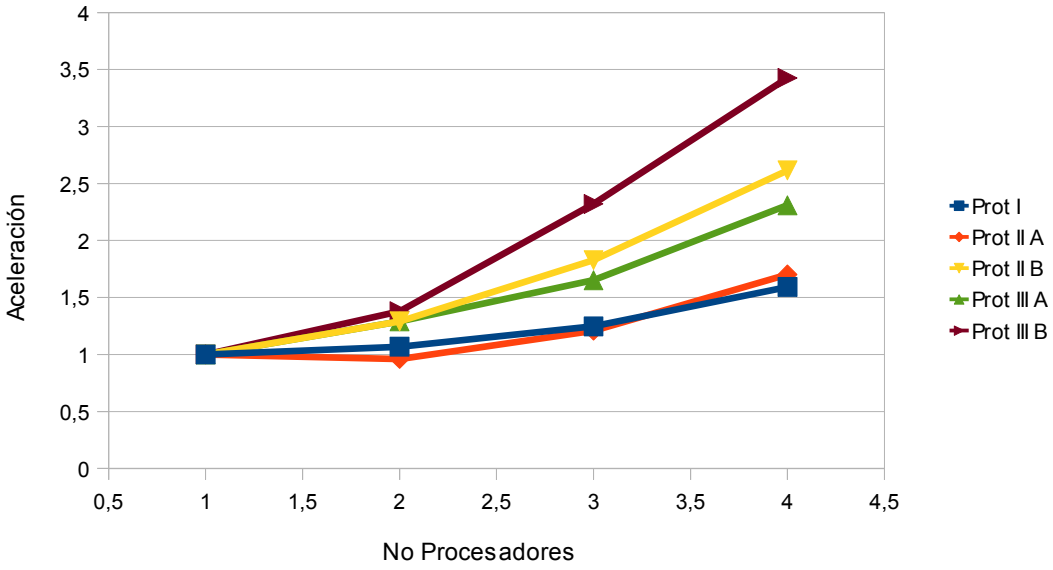
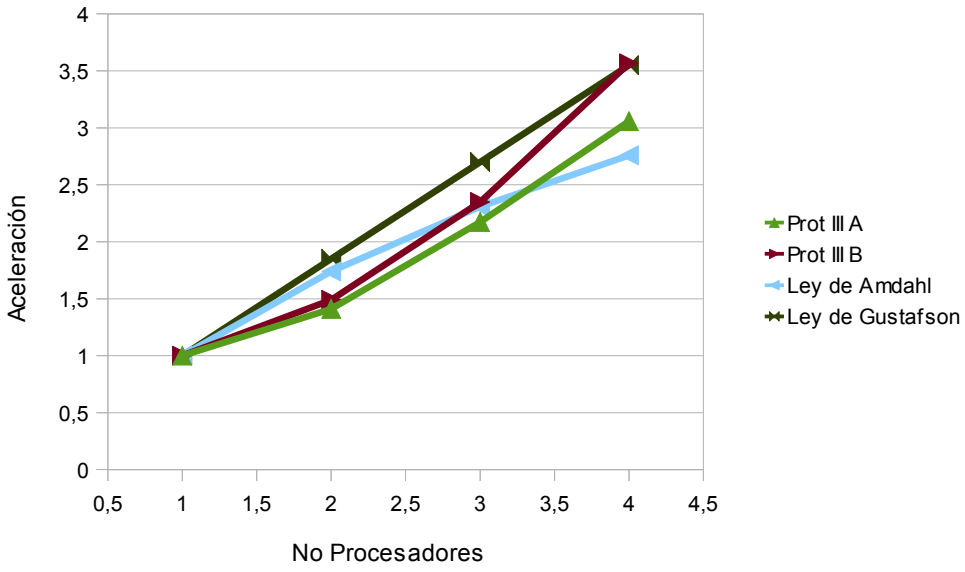


Imagen 7:

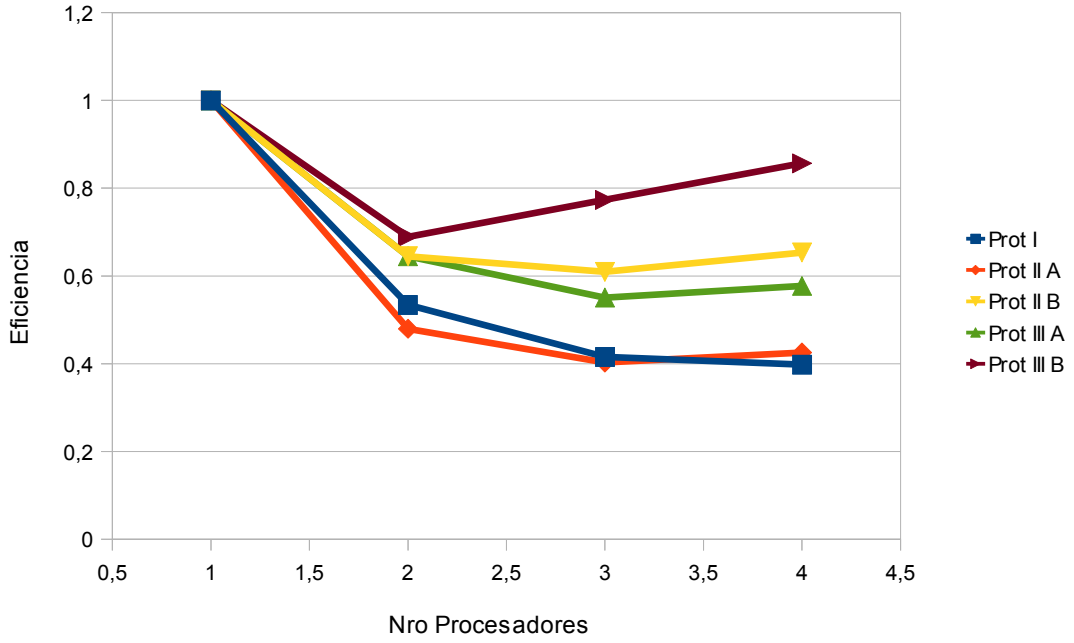
Gráfica 34. Resultados Aceleración Imagen 7



Gráfica 35. Resultados Aceleración, Ley de Amdahl, Ley de Gustafson, Imagen 7



Gráfica 36. Resultados Eficiencia Imagen 7



Gráfica 37. Resultados Eficiencia Teórica Imagen 7

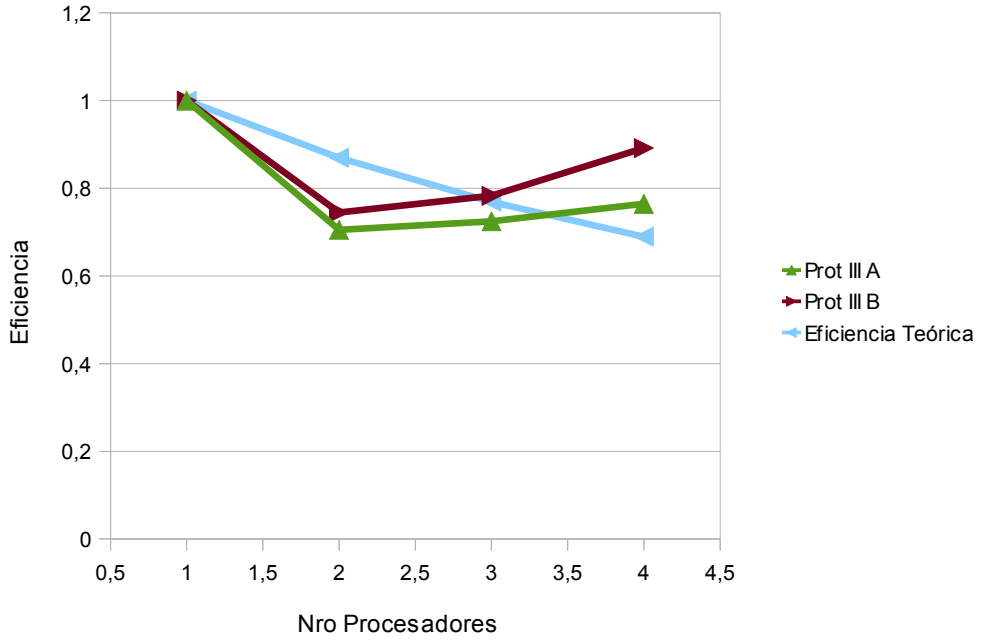
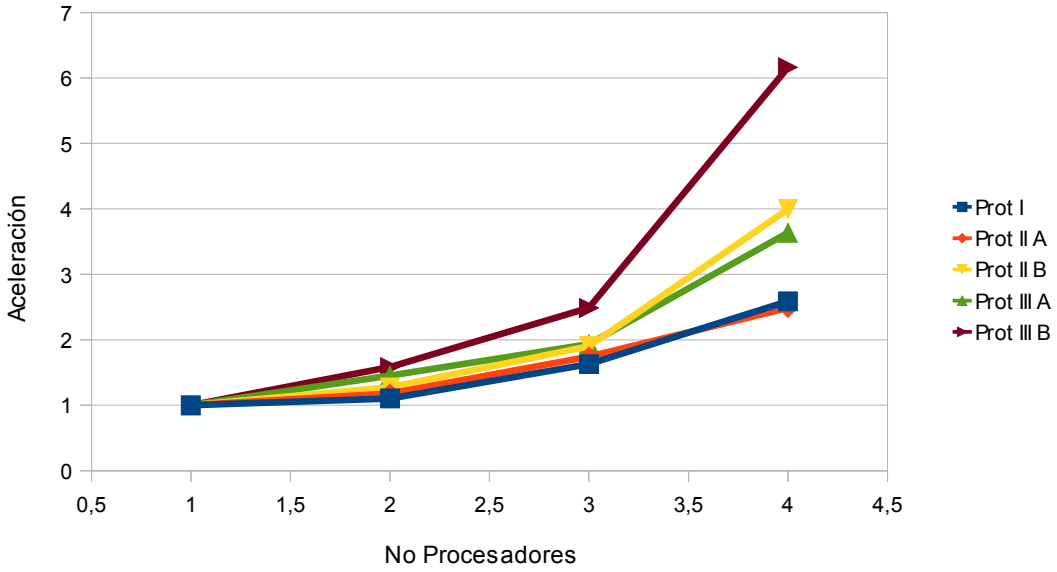
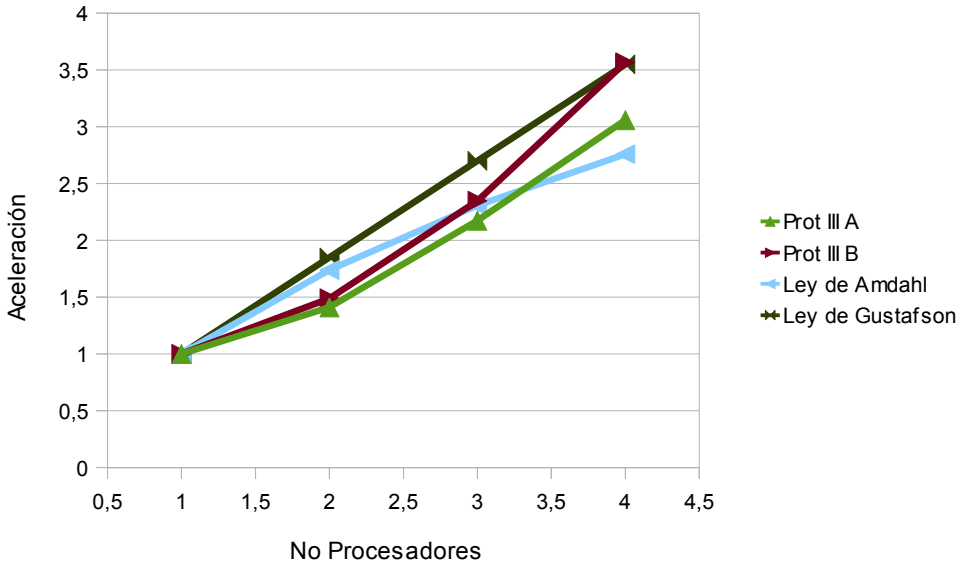


Imagen 8:

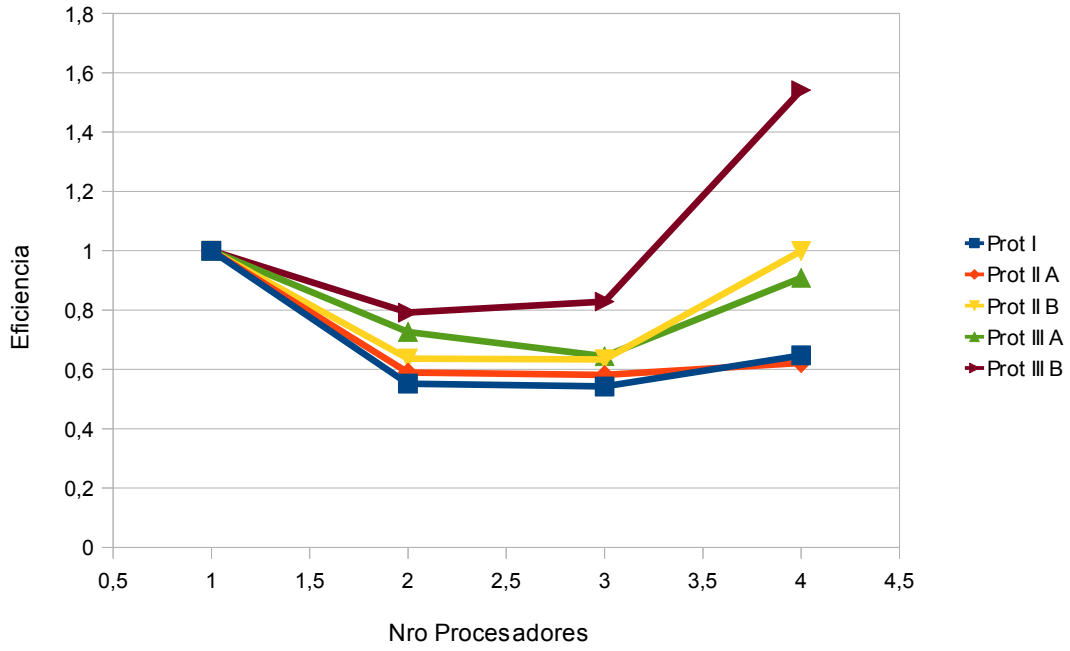
Gráfica 38. Resultados Aceleración Imagen 8



Gráfica 39. Resultados Aceleración, Ley de Amdahl, Ley de Gustafson, Imagen 8



Gráfica 40. Resultados Eficiencia Imagen 8



Gráfica 41. Resultados Eficiencia Teórica Imagen 8

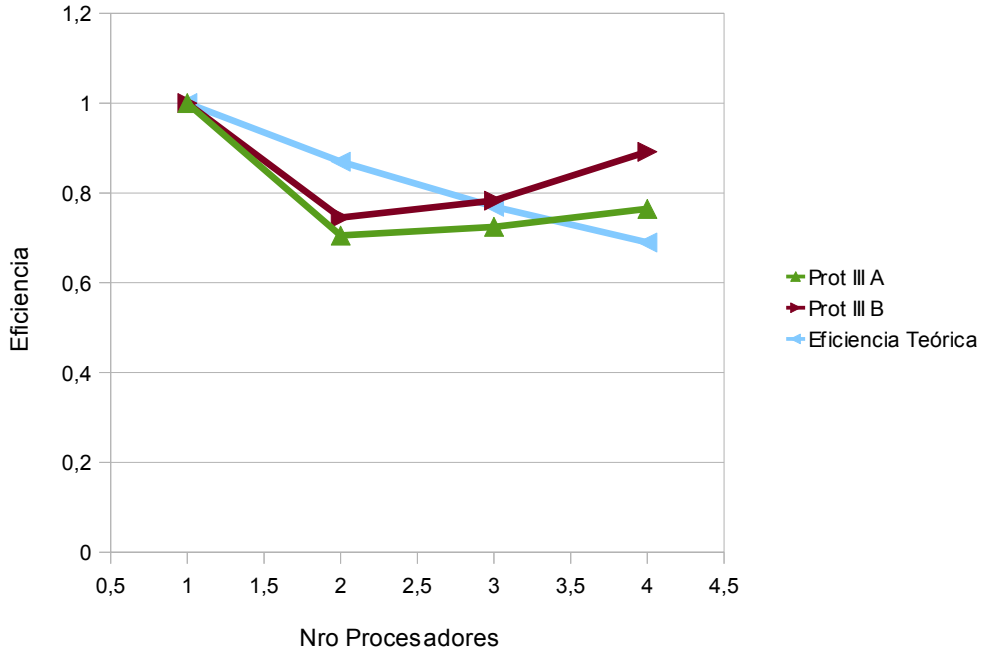
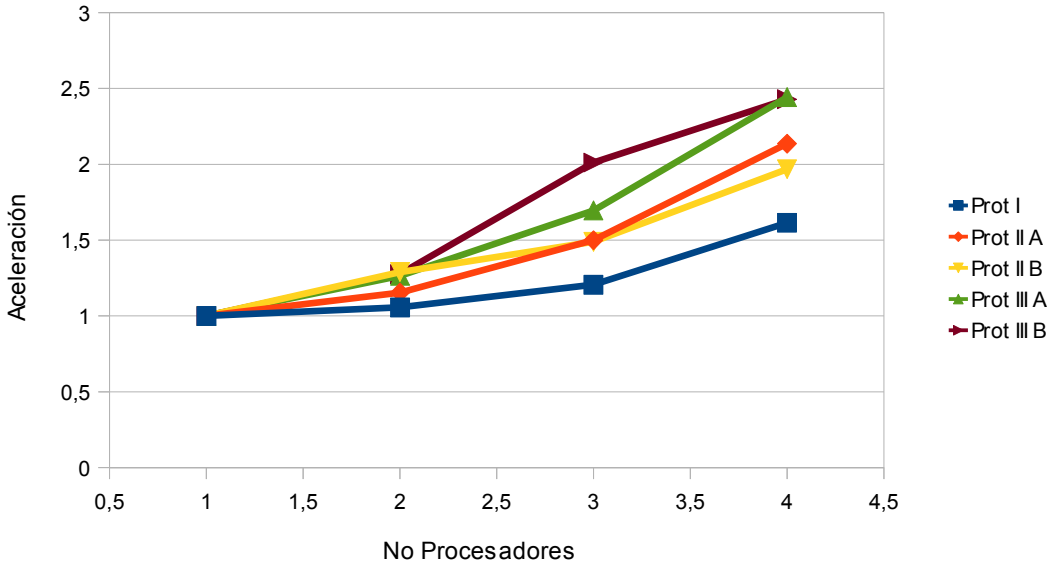
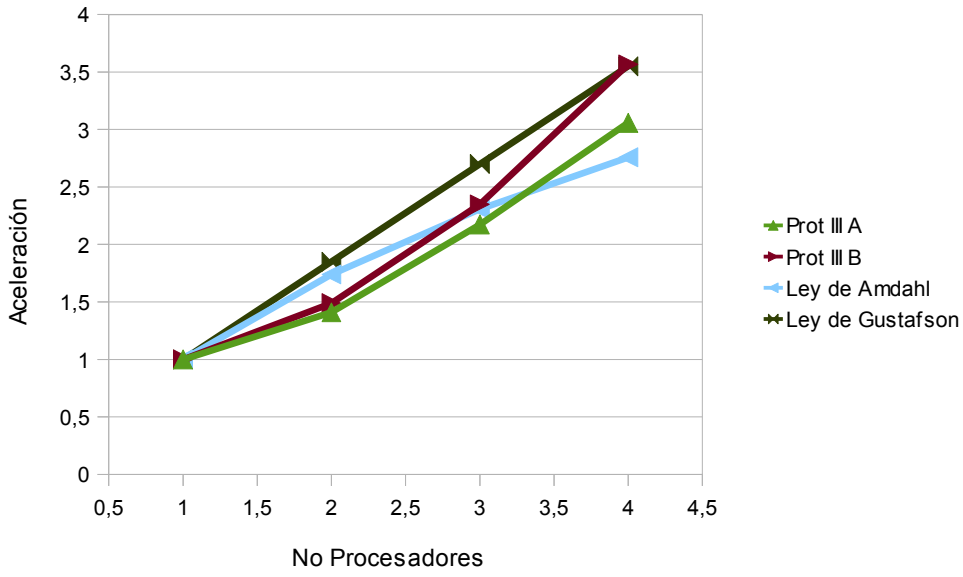


Imagen 9:

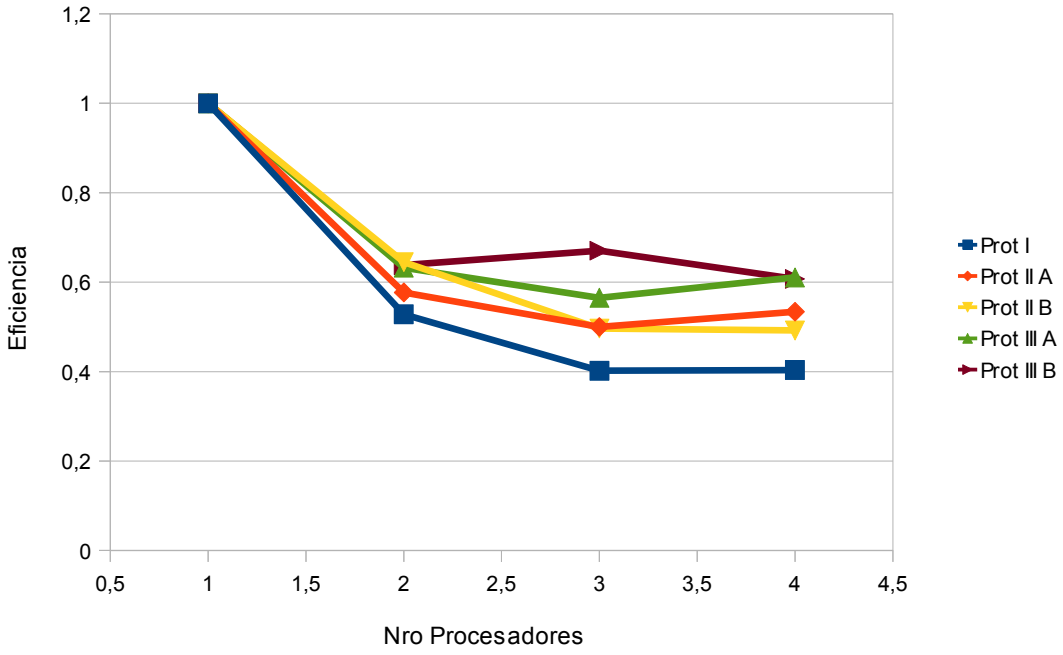
Gráfica 42. Resultados Aceleración Imagen 9



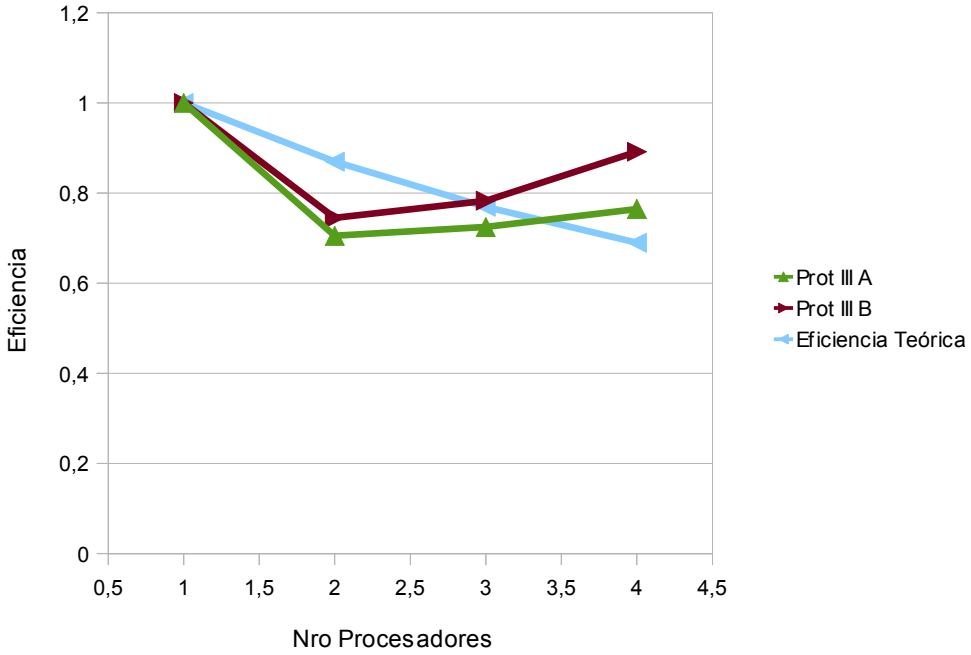
Gráfica 42. Resultados Aceleración Imagen 9



Gráfica 44. Resultados Eficiencia Imagen 9



Gráfica 45. Resultados Eficiencia Teórica Imagen 9



Se puede observar en la gráfica de aceleración de las imágenes 5, 6 y 9, que para el prototipo III B que había presentado los mejores tiempos de ejecución, en este caso no presenta muy buenos resultados en la aceleración y la eficiencia.

El resultado pesimista puede ser debido a que la ley de Amdahl transmite la realidad de que el rendimiento no aumenta por incrementarse la cantidad de nodos, ya que una pequeña parte no paralizable puede provocar una disminución del rendimiento en el sistema.

También podría deberse a que el volumen de cálculos a medida que se aumenta el número de nodos, no es fijo. Por lo cual seria conveniente evaluar el sistema con la ley de Gustafson, en un sistema de escalabilidad, para comprobar como se comporta la aceleración.

Por lo pronto y aunque la aceleración no es tan elevada, para el prototipo III, los resultados en tiempo de ejecución nos refleja un buen rendimiento del algoritmo.

Para las demás imágenes, los resultados de la aceleración y la eficiencia fueron muy buenos para el prototipo III, por lo cual se puede concluir que la paralelización del algoritmo fue eficiente.

7.2. Comunicaciones

La comunicación en el cluster se evalúa a través del ancho de banda y la latencia. Para evaluar estos valores se ejecuta un programa en paralelo que envíe 100 mensajes entre 2 nodos y se cambia el tamaño del mensaje obteniendo la siguiente información promedio: [33]

Tabla 8. Evaluación Comunicaciones

Cantidad Datos (K)	Ancho de Banda (MB/s)	Latencia (microseg)
50	1.88	69
100	1.91	56
500	1.9	38
1000	1.92	35

Se puede observar que a medida que se envían mayor cantidad de datos, la latencia disminuye. El ancho de banda se mantiene estable y la latencia varia según la cantidad de datos.

[33] [Rec07] Comunicación, pág 55

Capítulo 8

Conclusiones y Recomendaciones

Después de analizar los resultados del algoritmo, en este capítulo se presentan las conclusiones del trabajo realizado en el proyecto. Primero se concluye cual de los tres prototipos diseñados produjo mejores resultados y se estima el impacto que tuvo la estrategia de diseño del algoritmo. Luego se analiza si el desempeño alcanzado fue acorde al esperado. Continuando se presentan las conclusiones respecto a los objetivos planteados en el plan de proyecto y por último las recomendaciones para futuros trabajos.

8.1. Estrategia de Diseño

Se puede observar que los prototipos para los cuales se produjeron los mejores resultados fueron el Prototipo II y III, en los cuales se mejoró el sistema de comunicaciones y se introdujeron algoritmos de balance de carga.

La forma de identificar las regiones también afecto el rendimiento del algoritmo, se puede observar que las estrategias de identificación cíclica y aleatoria producen mejores resultados ya que no se genera una secuencia ordenada lo que produce desbalanceos de carga y mayor número de iteraciones.

En general, a medida que en los prototipos se va mejorando estas tres características, el rendimiento se va incrementando.

8.2. Desempeño Alcanzado

En el análisis del rendimiento se calcularon los valores reales de Aceleración y Eficiencia y los valores teóricos de ley de Amdahl, ley de Gustafson y Eficiencia en las nueve imágenes de prueba.

Obteniendo como resultado no tan buenos en la aceleración para las imágenes 5, 6 y 9, a pesar que en las pruebas de tiempo de ejecución fueron los mejores, lo cual se logra notar con los resultados de la ley de Amdahl. Por otra parte puede que la aceleración crezca a medida que aumenta el número de nodos, lo cual valdría evaluarlo con la Ley de Gustafson y así determinar si solo se trata de la visión pesimista de la ley de Amdahl. Para el resto de las imágenes la aceleración tuvo muy buenos resultados.

Por último los resultados de la eficiencia estuvieron muy cercanos a uno, lo que significa que los procesadores están siendo utilizados en una gran parte de su capacidad de procesamiento y esto es debido a que los algoritmos de balance de carga equilibran los procesos en los nodos.

Con los resultados expuestos bajo estas métricas de rendimiento puede concluirse que el desempeño del algoritmo fue satisfactorio.

Respecto al Hardware, la arquitectura paralela del cluster afectó el rendimiento del algoritmo ya que al compartir la memoria, los procesadores se demoran más accediendo a los datos, además que el cluster instalado no es de alto rendimiento.

8.3. Conclusiones Generales

Objetivo General

Implementar una versión en paralelo del algoritmo basado en el método de regiones Split & Merge para el proceso de segmentación de imágenes digitales de muestras Cervico-Uterinas.

Se diseñó y desarrolló un algoritmo paralelo Split & Merge para segmentar imágenes digitales, en este caso muestras cervicales. Se realizaron pruebas y análisis y se determinó que los factores más relevantes a tener en cuenta al momento de obtener un mayor rendimiento del algoritmo son el sistema de comunicaciones entre procesadores, los algoritmos de balance de carga y la forma de identificar las regiones.

Objetivos Específicos

- *Aplicar la metodología del diseño de algoritmos paralelos al algoritmo de crecimiento de regiones.*

Se aplicó el método de Ian Foster para diseño de algoritmos paralelos (Particionamiento, Comunicación, Aglomeración, Asignación). Para el particionamiento no se efectuó algún cambio, ya que es inherentemente paralela. En el sistema de comunicaciones se probaron varios modelos y el que mejor resultado generó fue la topología de grafo, ya que en esta comunicación se tiene definido que tareas se comunican entre sí entre procesadores adyacentes. En la aglomeración la forma de identificar las regiones para agruparse ayuda a balancear la carga en los procesadores y las formas que mejores resultados

produjeron fueron la cíclica y la aleatoria. En la asignación, antes de enviar las tareas fue necesario balancear la carga de procesos en los nodos, y después de cada iteración, para obtener mayor rendimiento.

- *Analizar y definir los recursos hardware y las herramientas software a utilizar en el proyecto para optimizar el tiempo de procesamiento.*

Se eligieron los recursos hardware tipo Cluster Linux y para la programación del algoritmo el lenguaje gcc y para el paso de mensajes OpenMPI, por la simplicidad de manejo de un Cluster Linux y la programación en gcc, brindando un ambiente propicio para la computación paralela para este tipo de algoritmo que no necesita de un ambiente más robusto.

- *Desarrollar un prototipo a partir de la implementación de una versión paralela del algoritmo de crecimiento de regiones Split & Merge.*

Se desarrollaron tres prototipos, los cuales se analizaron y se concluyó que el prototipo III obtuvo el mejor resultado en tiempo de ejecución, el cual se eligió como prototipo final.

- *Verificar y evaluar el resultado de la segmentación realizada por el algoritmo paralelo en imágenes médicas (muestras cervico-uterinas).*

Se eligió una cantidad de 9 imágenes de muestras cervicales, bajo el criterio de la cantidad de objetos que se deben segmentar. Las cuales se segmentaron con el algoritmo paralelo. Como resultados se obtuvieron para las imágenes 5, 6 y 9 la aceleración no fue muy alta como para el resto de imágenes para las cuales la aceleración y la eficiencia demostraron un buen rendimiento del algoritmo.

- *Validar los tiempos de ejecución entre la versión secuencial y la versión paralela.*

En el análisis de resultados se validaron los tiempos de las dos versiones mediante las métricas de aceleración y eficiencia para el prototipo final III.

8.4. Recomendaciones

Seria conveniente realizar pruebas en Clusters de alto rendimiento con más nodos para evaluar la escalabilidad del algoritmo con la ley de Gustafson, ya que el nivel de cálculos en el algoritmo aumenta a medida que se incrementan los procesadores, y de esta forma se podrían obtener más claramente los resultados sobre el rendimiento del algoritmo.

Bibliografía

Libros

[Bou010] Bourne, Roger. “Fundamentals of Digital Image in Medicine”. Springer London Dordrecht Heidelberg New York. 2010.

[BT02] Bertsekas, D. Tsitsiklis, J. Parallel and Distribuited computation Numerical Methods, Prentice Hall, 2002.

[Cast96] Castleman, Kenneth. “Digital Image Processing”. Upper Saddle River, NJ. Editorial Prentice Hall. 1996.

[DS98] Down, Kevin. Severance, Charles. “High Performance Computing”. United States of America, Editorial O'Reilly. 1998.

[Fos95] Foster, Ian. “Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering”. Addison-Wesley Longman Publishing Co., Inc. 1995.

[GW02] Gonzalez, R.C. Woods, R.E. “Digital Image Processing”, 2nd Edition, Prentice-Hall, Inc., Upper Saddle River, NJ. 2002.

[KK01] Karniadakis, George. Kirby, Robert. Parallel Scientific Computing in C++ and MPI. Ed. Cambridge University. 2001.

[Pra07] Pratt, William. “Digital Image Processing”, Fourth Edition, John Wiley & Sons, Inc., 2007.

Artículos

[BD] Brun, Luc. Domenger , Jean. “A new split and merge algorithm based on Discrete Map ”. LaBRI, Universite Bordeaux I . 351, Cours de la liberation 33405 Talence cedex, France.

[FPA+08] Faruquzzaman, A.B.M. Paiker, N.R. Arafat, J. Karim, Z. Ameer Ali, M. "Object segmentation based on split and merge algorithm," TENCON 2008 - 2008. IEEE Region 10 Conference , vol., no., pp.1-4, 19-21, Nov. 2008.

[HP74] Horowitz, S.L. and Pavlidis, T. “Picture segmentation by a directed split-and-merge procedure”. Proceedings, 2nd International J. C. on Pattern Recognition, Copenhagen, 424–433. 1974.

[HP76] Horowitz, S. L. and Pavlidis, T. “Picture Segmentation by a Tree Traversal Algorithm”. J. ACM 23, 2 (Apr. 1976), 368-388. 1976.

[JCSW92] Jin-Shin Chou, Chin-Tu Chen, Shiuh-Yung J. Chen and Wei-Chung Lin, "Parallel implementation of an adaptive split-and-merge method for medical image segmentation", Proc. SPIE 1652, 62 (1992).

[KG08] Kelkar, D. Gupta, S. “Improved Quadtree Method for Split Merge Image Segmentation”. In: Proceedings, First international Conference on Emerging Trends in Engineering and Technology - Volume 00 (July 16 - 18, 2008). ICETET. IEEE Computer Society, Washington, DC, 44-47. 2008.

[Mar05] Martinez, V. et al. “Computational model for squamous cells characterization during cervical smear cytology” . Rev. Colomb. Biotecnol. Vol. VII No. 2 Diciembre 2005, 35-46. 2005.

[Mar07a] Martinez, V. et al. 2007. “Tecnología Grid para la Detección de Cáncer de Mama y Cuello Uterino por Medio de Procesamiento de Imágenes”. CLCAR 2007. Colombia, 13-18 Agosto 2007, 318-324.

[MGG99] Montoya, M.D.G.; Gil, C.; Garcia, I., "Load balancing for a class of irregular and dynamic problems: region growing image segmentation algorithms," Parallel and Distributed Processing. Proceedings of the Seventh Euromicro Workshop on , vol., no.,

pp.163-169, 3-5, Feb 1999.

[MGG00] Montoya, M.G. Gil, C. García I. “Implementation of a Region Growing Algorithm on Multicomputers: Analysis of the Work Load Balance.” In Proceedings of the Eighteen IASTED International Conference. Applied Informatics, pages 695-701, Innsbruck (Austria), February 2000.

[MGG03] Montoya, M. D. Gil, C. and García, I. “The load unbalancing problem for region growing image segmentation algorithms”. J. Parallel Distrib. Comput. 63, 4 (Apr. 2003), 387-395. 2003.

[PSR02] Pichel, J.C. Singh, D.E. Rivera, F.F. "Algoritmo paralelo de segmentación de imágenes basado en el crecimiento desacoplado de regiones", Conferencia Iberoamericana en Sistemas, Cibernética e Informática, Orlando, Florida, USA, 2002.

[PSR07] Pichel, J.C. Singh, D.E. Rivera, F.F. "A parallel framework for image segmentation using region based techniques", Vision systems. Segmentation and pattern recognition, Vienna. Austria, I-Tech Education and Publishing, pp. 81-98, 2007.

[RGO04] Redondo, J.L. García, I. Ortigosa, P.M. Paralelización de un Algoritmo Multimodal. In I. García, L.G. Casado, V.G. Ruiz, and J.J. Fernández, editors, Actas de las XV Jornadas de Paralelismo. Computación de Altas prestaciones, pages 414-419, Almeria, Septiembre 2004.

[SHR00] Singh, D.E. Heras, D.B. Rivera, F.F. "Algoritmo paralelo de segmentación basado en el crecimiento de regiones a partir de semillas", Revista Electrónica de Visión por Computador (REVC), 2000.

[TB90] Tyagi, A. Bayoumi, M. "Systolic array implementation of image segmentation by a directed split and merge procedure," Pattern Recognition, 1990. Proceedings., 10th International Conference on , vol.ii, no., pp.491-493 vol.2, 16-21, Jun 1990.

[Til88] Tilton, J.C., "Image segmentation by iterative parallel region growing with applications to data compression and image analysis," Frontiers of Massively Parallel Computation. Proceedings., 2nd Symposium on the Frontiers of , vol., no., pp.357-360, 10-12 Oct 1988.

Tesis

[A+06a] Anaya, S. et al. " Mesoft 1.0: Prototipo Software para la Caracterización de Células Mesoteliales noTtumorales Malignas en Citología de Líquido Pleural". Trabajo de Grado. UIS. 2006.

[A+06b] Amaya, M.L. et al. "Contribución al estudio de las Células Escamosas de Citologías Cérvico Uterinas que Presentan Cambios por Virus de Papilona Humano (VPH), utilizando Tratamiento Digital de Imágenes". Trabajo de Grado. UIS. 2006.

[Her05] Hernández, Miguel. "El Procesamiento Distribuido y su aplicación al Tratamiento de Imágenes ". Universidad Politécnica de Madrid. 2005

[Hor06] Hortua, D. "Diseño e Implementación de la Versión Paralela del Programa para el Modelamiento de Propagaci;on de Ondas Elasticas 2d en Cuerpos Sólidos Usando el Método de Diferencias Finitas Ecoelas2d ". Trabajo de Grado. UIS. 2006.

[Lak89] Lakshman, Prabhashankar . "Parallel implementation of the split and merge algorithm on the hypercube machine". Thesis Master of Science. Faculty of the College of Engineering and Technology, Ohio University . 1989.

[Mar04] Martinez, V. "Implementación de un Modelo Computacional para Clasificación normal – displásica de las Células Escamosas de Citologías Cérvico Uterinas ". Trabajo de Grado. UIS. 2004.

[Mar07b] Martinez, V. "Modelo Computacional para Caracterización de células endocervicales". Trabajo de Maestria. UIS. 2007. 2007.

[N+06] Niño, D. et al. "Contribución al estudio de las células escamosas de citologías cérvico uterinas que presentan cambio por ASCUS, por medio de tratamiento digital de imágenes". Trabajo de Grado. UIS. 2006.

[Pic02] Pichel, Juan. "Parallel Algorithm of Segmentation Based on Region Grouping in Over-Segmentated Images". Tesis de Maestria. 2002.

[Rec07] Recamann, H. "Desarrollo e Implementación de Código en Paralelo de Dinámica Molecular para el Estudio de Zeolitas". Trabajo de Maestria. UIS. 2007.

[AS] Alfaro, Analí. Sipirán, Iván. " Diseño de un Algoritmo de Segmentación de Imágenes

aplicando el Funcional Mumford-Shah para mejorar el desempeño de los Algoritmos Clásicos de Segmentación". Universidad Nacional de Trujillo.

Documentos

[ACCP04] Alliance for Cervical Cancer Prevention (ACCP). "Planning and Implementing Cervical Cancer Prevention and Control Programs: A Manual for Managers". Seattle: ACCP. 2004.

[ARR01] Alvarez M, Rivas M. and Rukoz M. "Segmentación de Imágenes Biomédicas Mediante el Crecimiento de Regiones". Acta Científica Venezolana, 52: 192–198, 2001.

[HH] Hidrobo, Francisco; Hoeber, Herbert. "Introducción a MPI".

[Unab02] Universidad Autónoma de Bucaramanga. 2002. "Registro de población de cáncer en el área metropolitana de Bucaramanga". 2000-2001.

[Seo03] Seol , Eunyoung. "Running C/C++ Program in parallel using MPI". 2003.

Glosario

Adquisición: etapa del procesamiento digital de imágenes que permite obtener una imagen digital de un objeto, a partir de una escena.

Algoritmo Paralelo: algoritmo que puede ser ejecutado por partes en el mismo instante de tiempo por varias unidades de procesamiento, para finalmente unir todas las partes y obtener el resultado correcto.

Ancho de Banda: Cantidad en bytes de datos que se pueden transmitir en una unidad de tiempo.

Binarización: procedimiento para convertir una imagen que se encuentra en el formato de color verdadero, o en escala de grises, a un formato de dos colores (Blanco y Negro).

Células Endocervicales: células encontradas en una muestra de citología cérvico uterina. Cumplen una función de protección y lubricación.

Células Escamosas: células encontradas en una muestra de citología cérvico uterina. De acuerdo con su etapa de maduración se clasifican en: basales, parabasales, intermedias y superficiales.

Cérvix: Llamado también cuello uterino. Porción del útero que se une con la vagina.

Citología Cérvico Uterina: prueba de tamizaje realizada a las mujeres, que consiste en obtener una muestra adecuada de la unión exoendocervical y del endocérvix.

Cluster de Computadoras: conjuntos o conglomerados de computadoras construidos mediante la utilización de componentes de hardware comunes y que se comportan como si fuesen una única computadora.

Computación de Alto Rendimiento: herramienta muy importante en el desarrollo de simulaciones computacionales a problemas complejos. Para lograr este objetivo, la computación de alto rendimiento se apoya en tecnologías computacionales como los clusters, supercomputadores o mediante el uso de la computación paralela.

Computación Distribuida: es un nuevo modelo para resolver problemas de computación masiva utilizando un gran número de computadoras organizadas en racimos incrustados en una infraestructura de telecomunicaciones distribuida.

Computación Grid: es una tecnología innovadora que permite utilizar de forma coordinada todo tipo de recursos (entre ellos cómputo, almacenamiento y aplicaciones específicas) que no están sujetos a un control centralizado. En este sentido es una nueva forma de computación distribuida, en la cual los recursos pueden ser heterogéneos (diferentes arquitecturas, supercomputadores, clusters...) y se encuentran conectados mediante redes de área extensa (por ejemplo Internet).

Computación Paralela: es una técnica de programación en la que muchas instrucciones se ejecutan simultáneamente. Se basa en el principio de que los problemas grandes se pueden dividir en partes más pequeñas que pueden

resolverse de forma concurrente ("en paralelo"). Existen varios tipos de computación paralela: paralelismo a nivel de bit, paralelismo a nivel de instrucción, paralelismo de datos y paralelismo de tareas.

Digitalizador: dispositivo electrónico que permite convertir una señal análoga a un formato digital, por medio de la discretización.

Firma de Núcleo: función unidimensional del contorno nuclear.

Histograma: gráfico utilizado para la representación de la cada uno de los niveles de color.

Latencia: Tiempo necesario para que un paquete de información viaje desde la fuente hasta su destino

Lesión Intraepitelial Escamosa: término acuñado por el sistema Bethesda para referirse a un cambio en el tejido del cérvix, a veces se usa como otra palabra para tumor.

Ley de Amdahl: Define la aceleración potencial de un algoritmo ejecutado en una plataforma paralela.

Ley de Gustafson: Establece que cualquier problema suficientemente grande puede ser eficientemente paralelizado, ofrece una visión positiva de las ventajas del procesamiento paralelo, al contrario de la ley de Amdahl.

Menssaje Passing Interfase: es un estándar que define la sintaxis y la semántica de las funciones contenidas en una biblioteca de paso de mensajes diseñada para ser usada en programas que exploten la existencia de múltiples procesadores.

MIMD: Multiple instruction multiple data

MISD: Multiple instruction single data

Modelo de Programación Paralela: Conjunto de tecnologías software que permiten expresar algoritmos paralelos para implantar aplicaciones en la arquitectura adecuada.

Neoplasia: sinónimo de tumor, en el sentido de cualquier proceso que curse con proliferación celular excesiva, se aplica generalmente para tumores malignos.

Neoplasia Intraepitelial Cervical: crecimiento anormal que se origina en el cérvix, causado por una sola célula alterada.

Nivel de color: valor asociado con la cantidad de luz detectada por un dispositivo de adquisición. Depende del número de bits empleado, para 8 bits se encuentra entre el intervalo [0, 255].

Preprocesamiento: etapa del procesamiento digital de imágenes que permite mejorar las características de una imagen digital.

PVM: Máquina Virtual Paralela (conocida como PVM por sus siglas en inglés de *Parallel Virtual Machine*) es una biblioteca para el cómputo paralelo en un sistema distribuido de computadoras. Está diseñado para permitir que una red de computadoras heterogénea comparta sus recursos de cómputo (como el procesador y la memoria RAM) con el fin de aprovechar esto para disminuir el tiempo de ejecución de un programa al distribuir la carga de trabajo en varias computadoras.

Reconocimiento: etapa del procesamiento digital de imágenes que permite la clasificación de los objetos presentes en una escena.

Segmentación: etapa del procesamiento digital de imágenes que permite la separación de los objetos presentes en una imagen.

SIMD: Single instruction multiple data

SISD: Single Instruction Single Data

Sistema Basado en el Conocimiento: sistemas informáticos que involucran conocimiento propio de especialistas.

Super Computación: Hace referencia a computadoras con capacidades muy superiores a las de otras máquinas disponibles.

Taxonomia de Flynn: Distingue las arquitecturas de multiprocesamiento de acuerdo a las instrucciones y datos, cada uno de estos criterios puede tomar dos estados, sencillo o múltiple.

Umbralización: técnica de segmentación que consiste en binarizar una imagen, tomando como referencia un nivel del histograma.

VPH: virus de Papiloma Humano, relacionado ampliamente con el cáncer de cérvix.

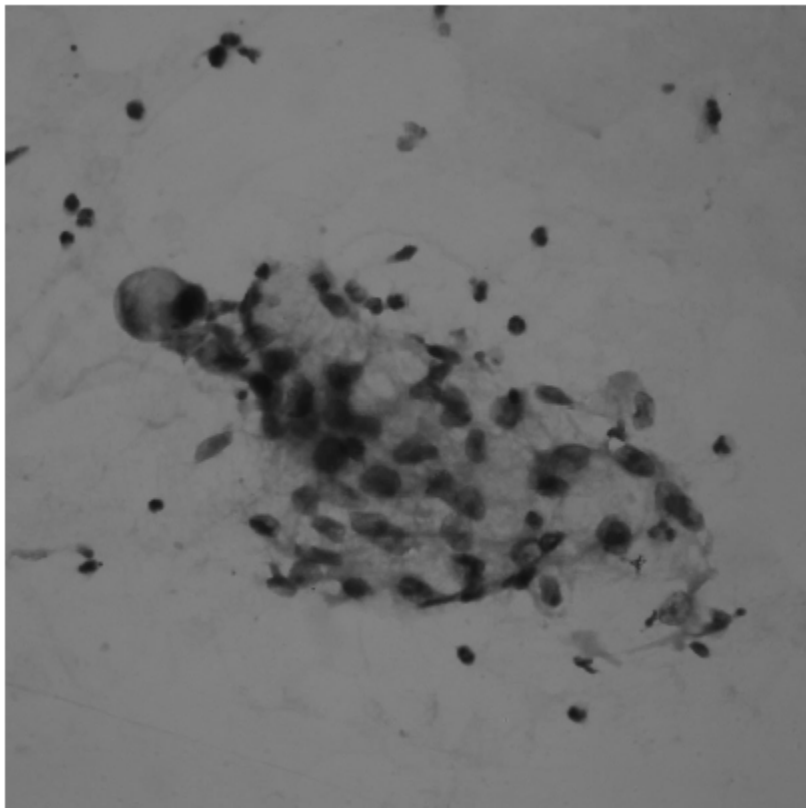
Apéndices

Apéndice A

Ejemplos de Imágenes Segmentadas

El proceso de segmentación con el algoritmo paralelo fue probado en dos imágenes de muestras células cervicales, sobre las dos versiones del prototipo III, porque dadas las conclusiones produjo los mejores resultados en tiempo de ejecución, a continuación se presentan los resultados para cada una de las imágenes de la salida que genera el algoritmo.

Imagen 1 :



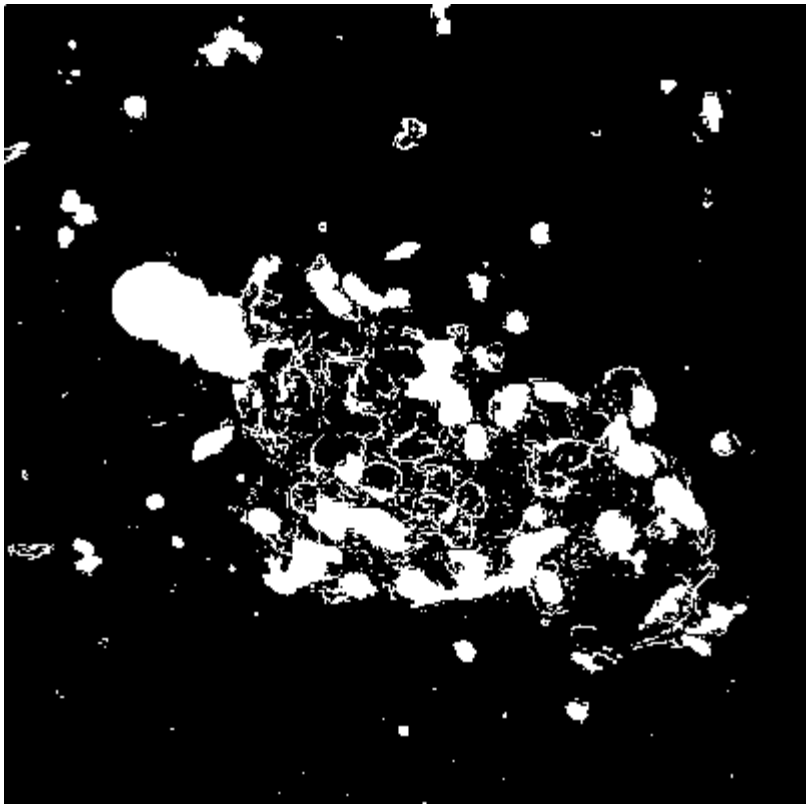
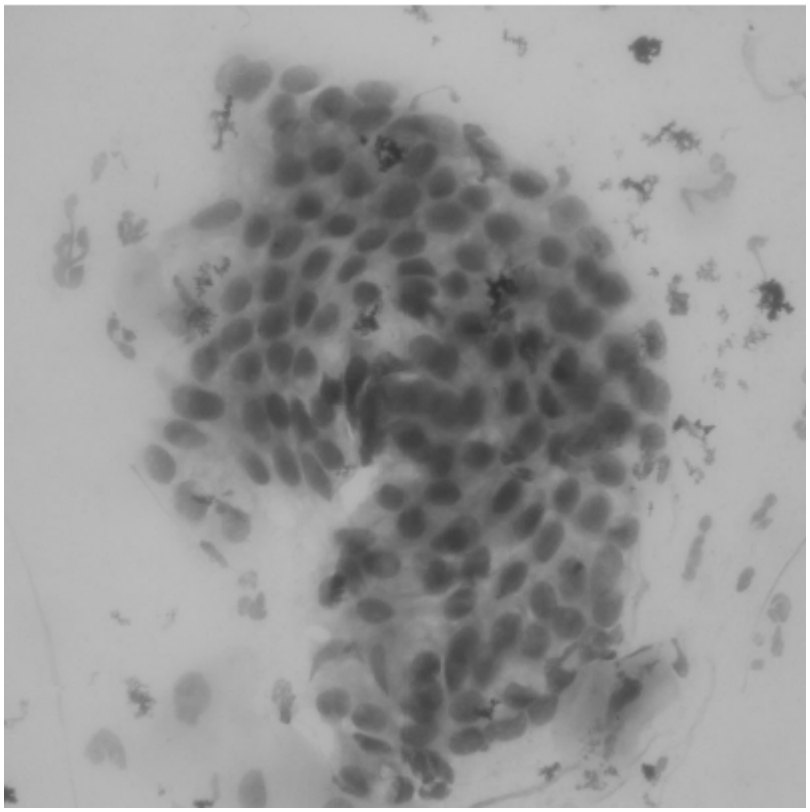


Imagen 2 :



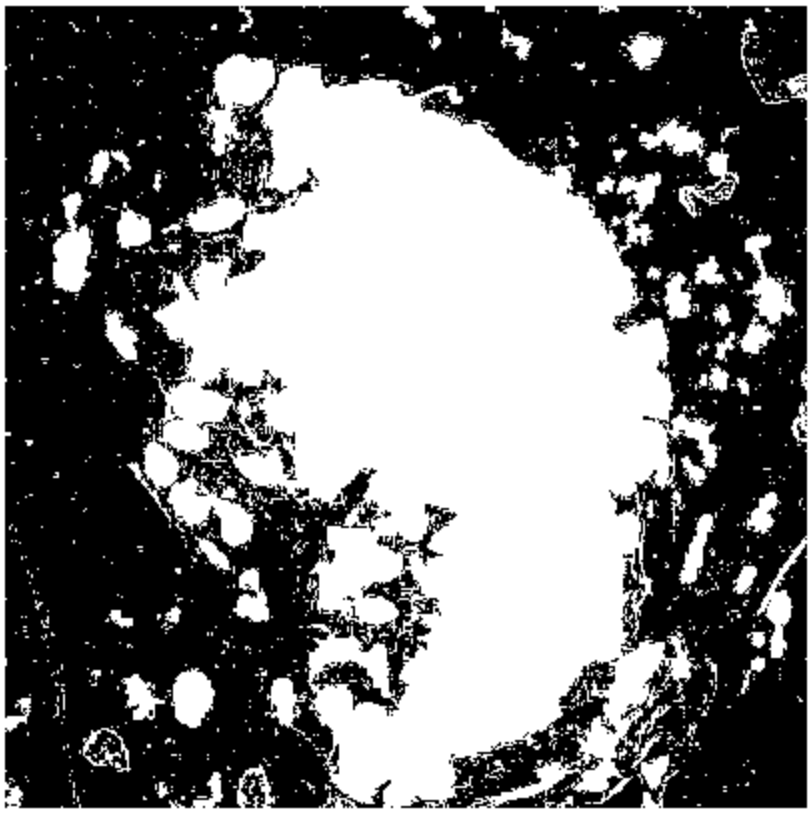
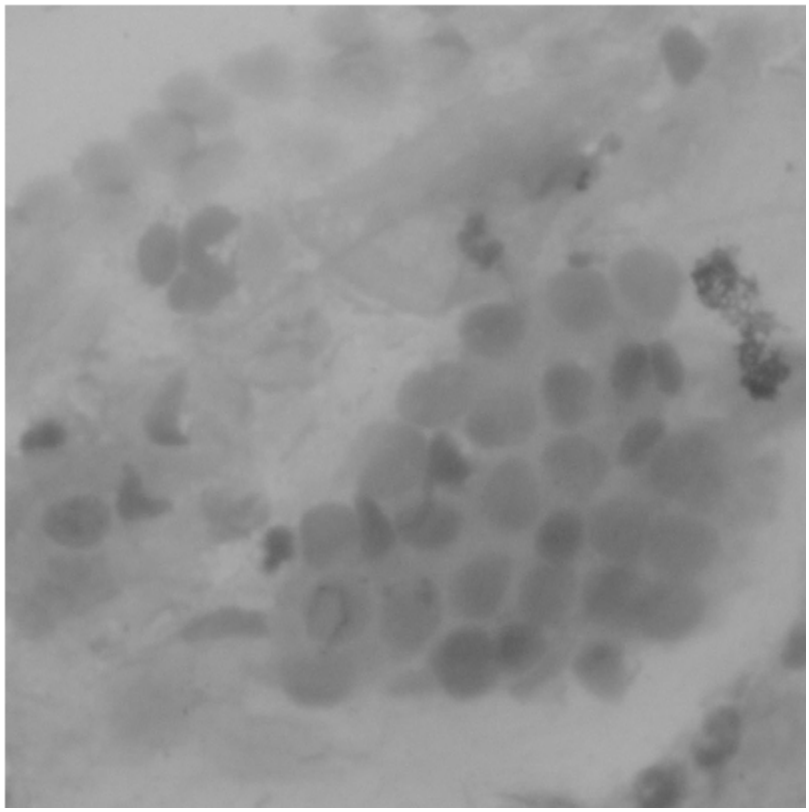


Imagen 3 :



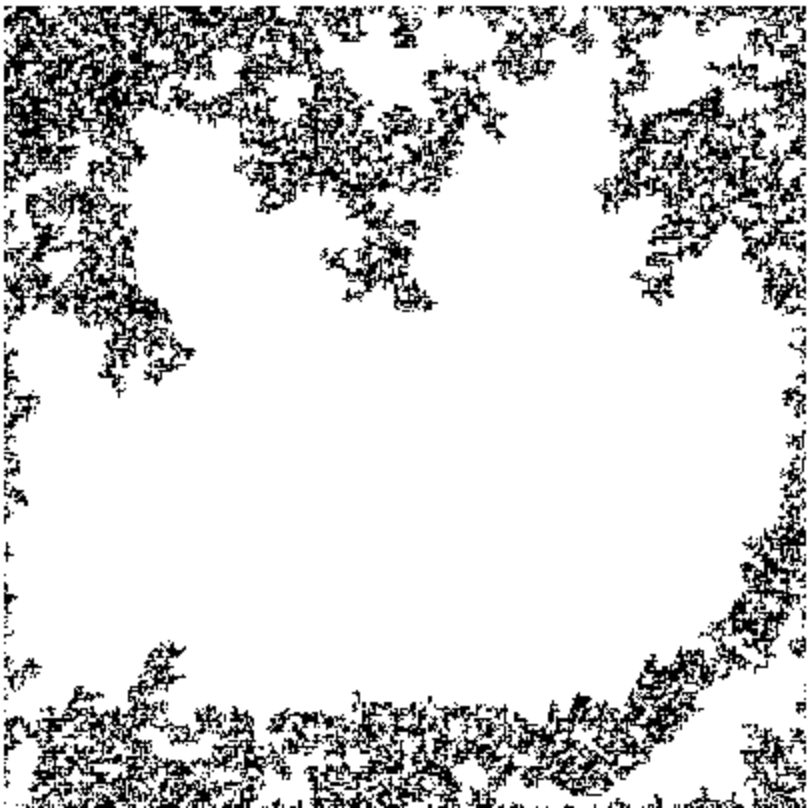
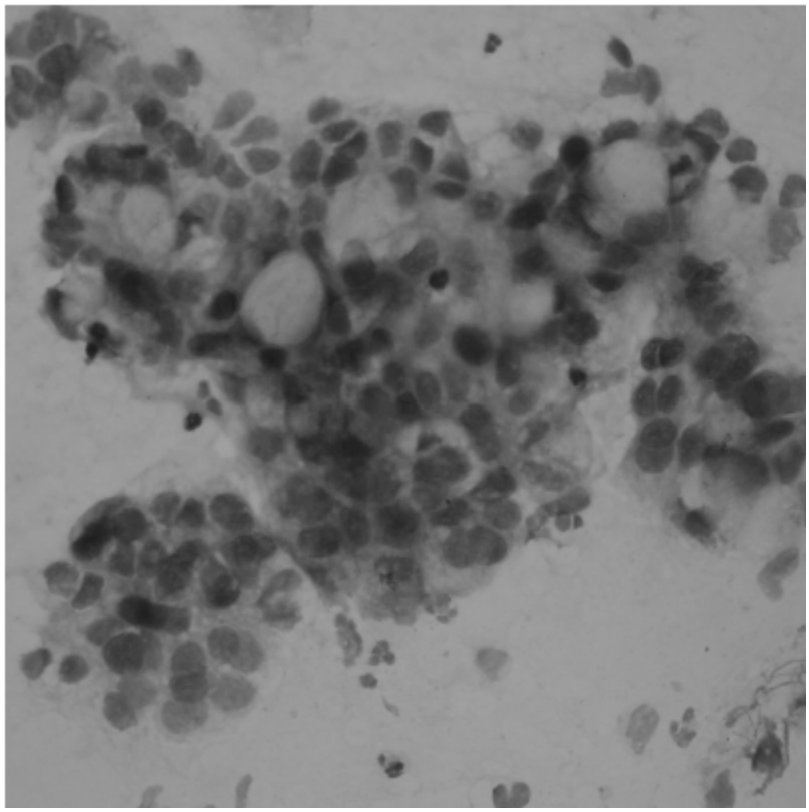


Imagen 4 :



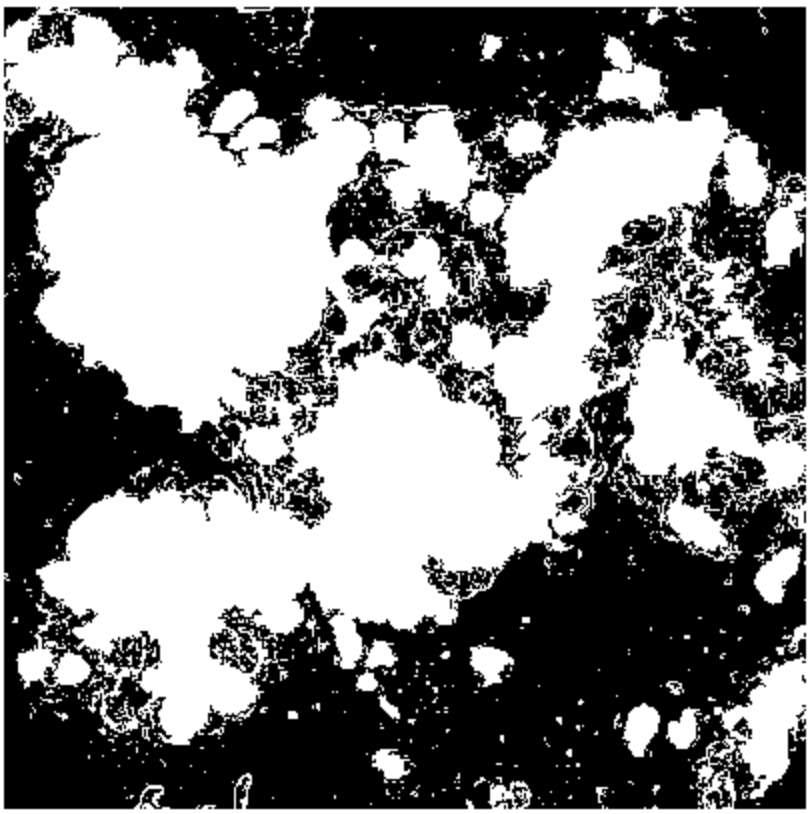
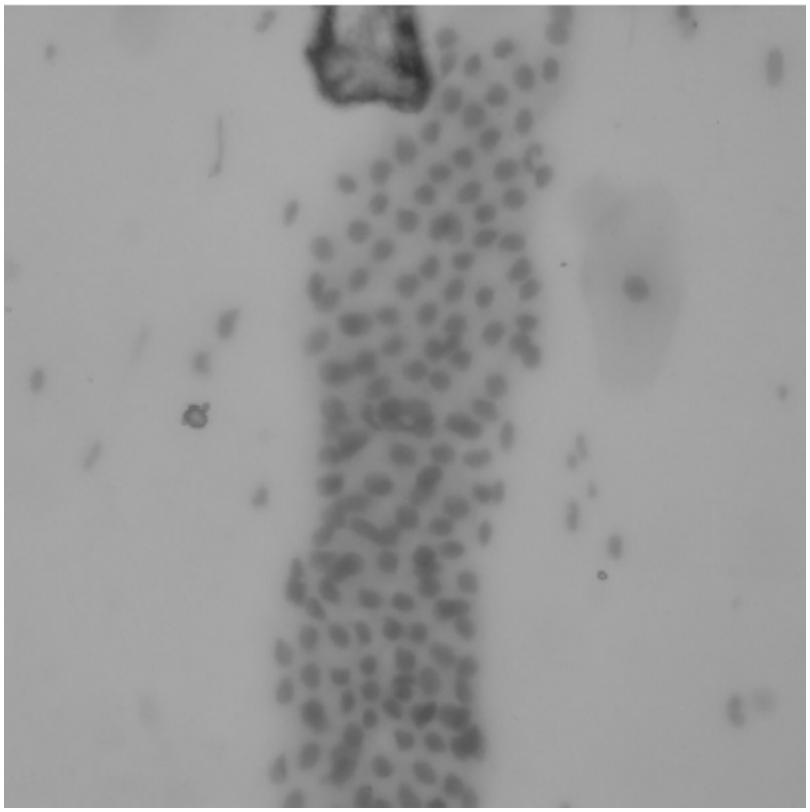


Imagen 5 :



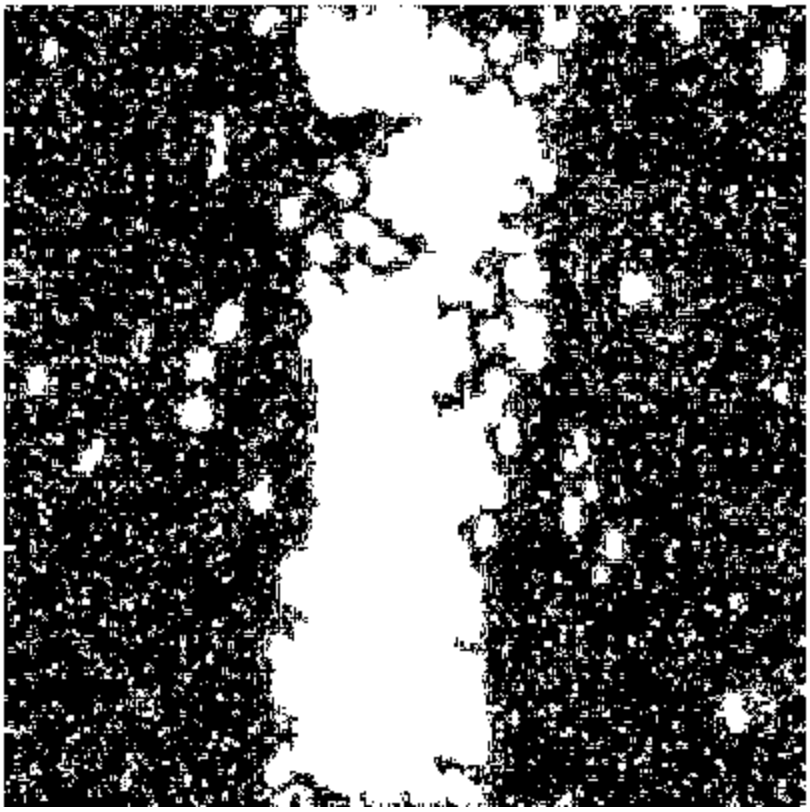
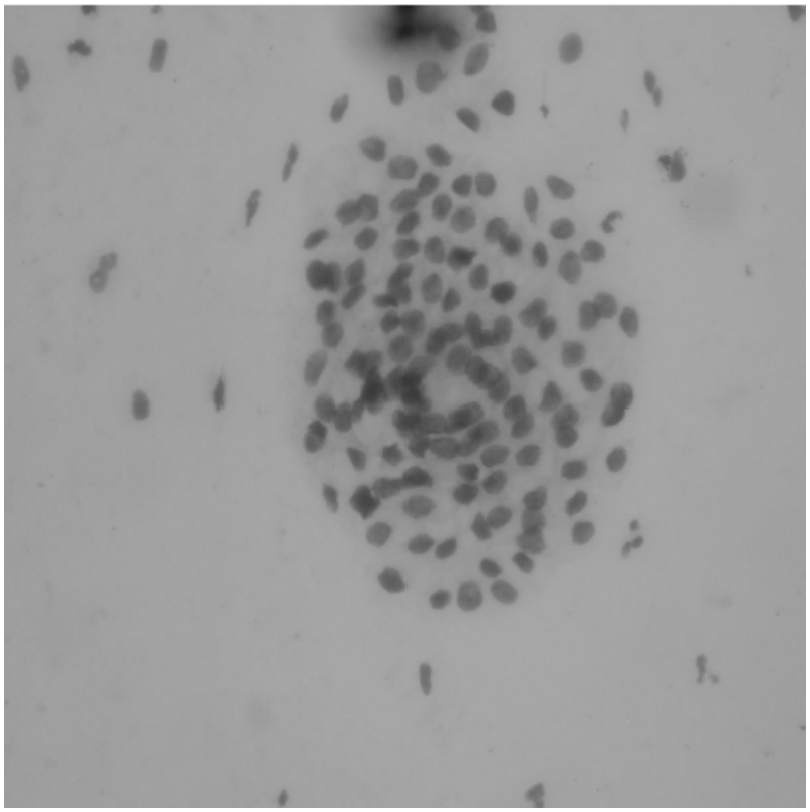


Imagen 6 :



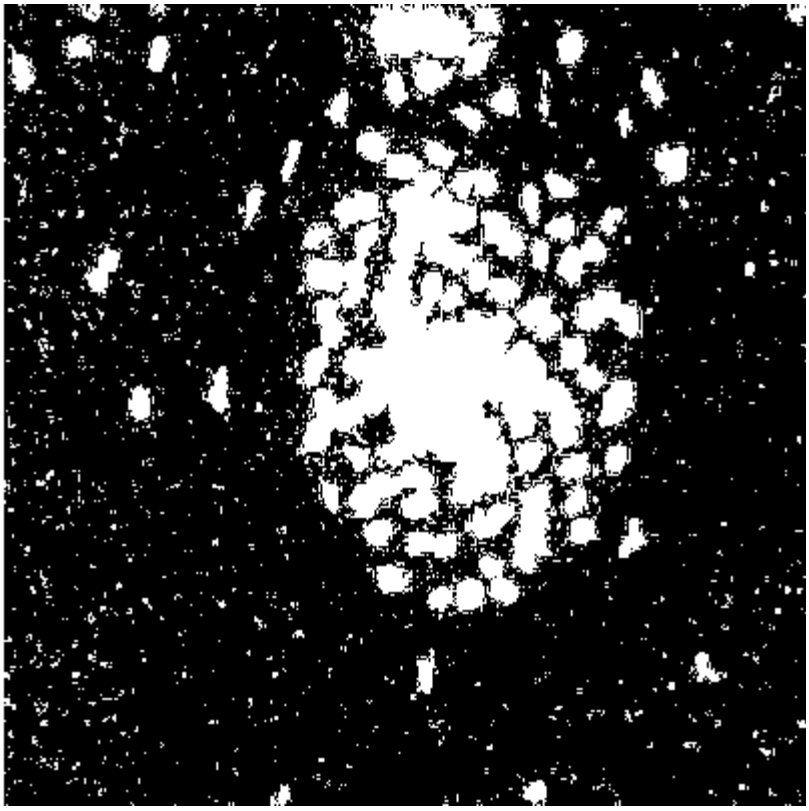
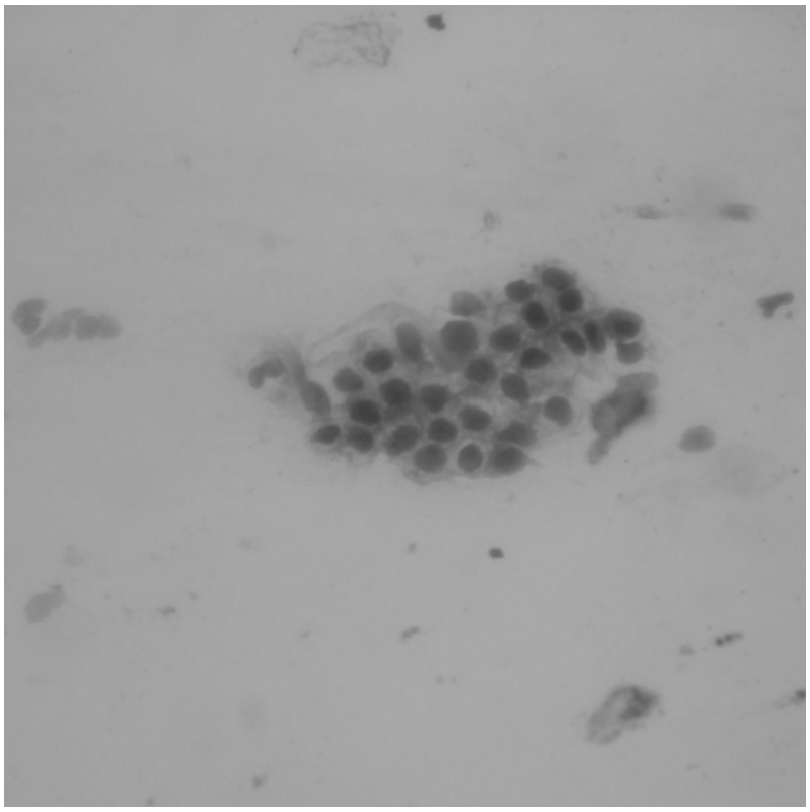


Imagen 7 :



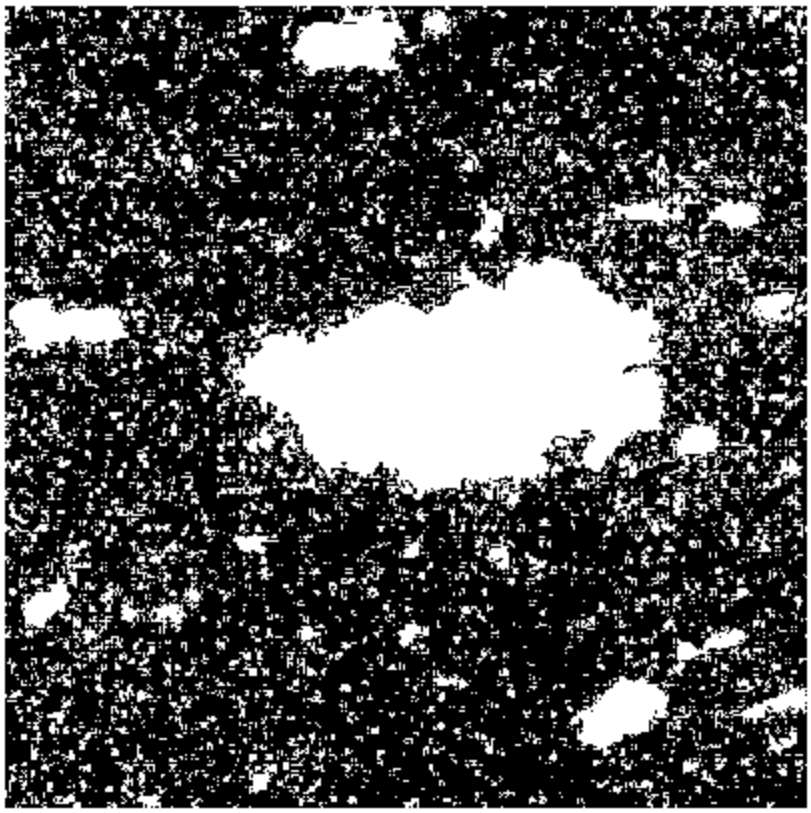
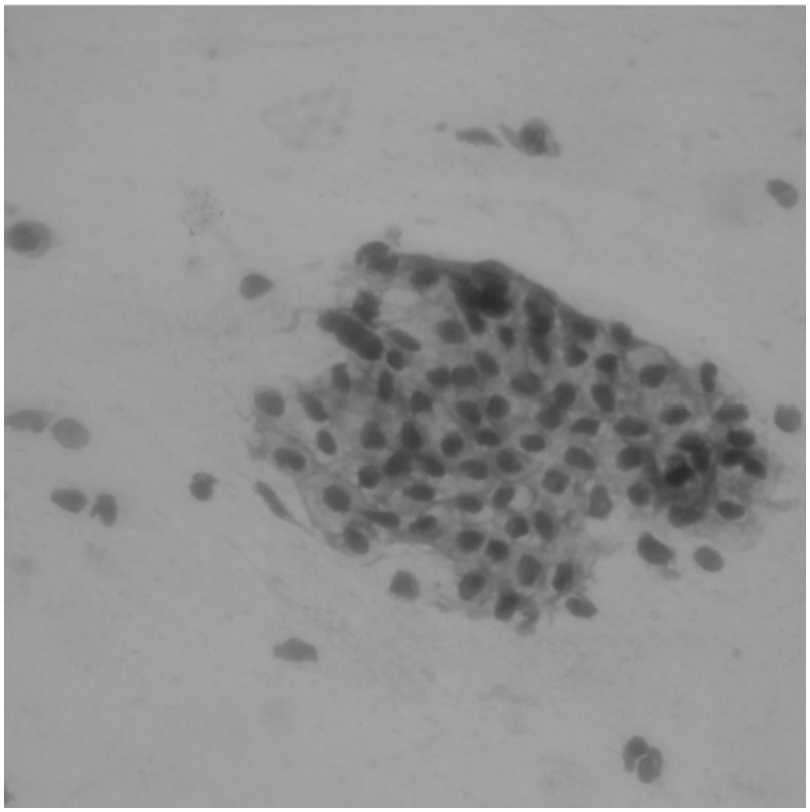


Imagen 8 :



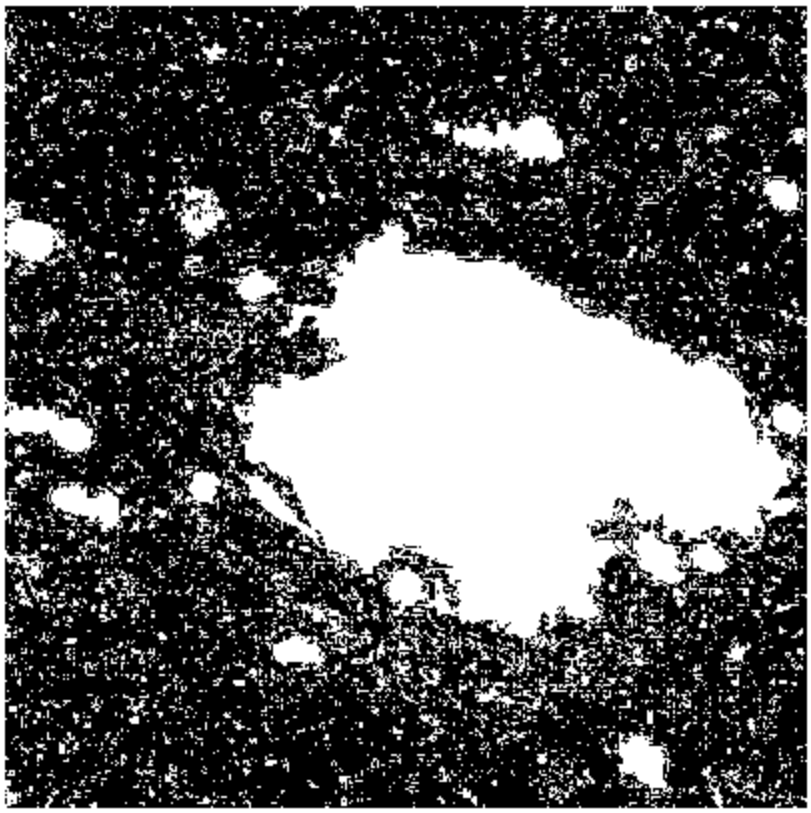
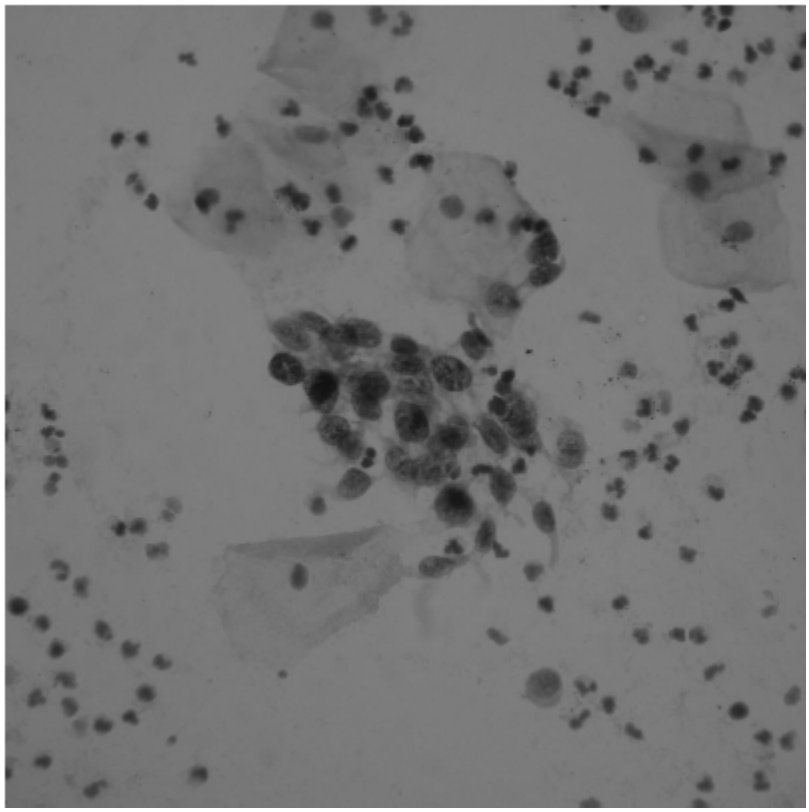
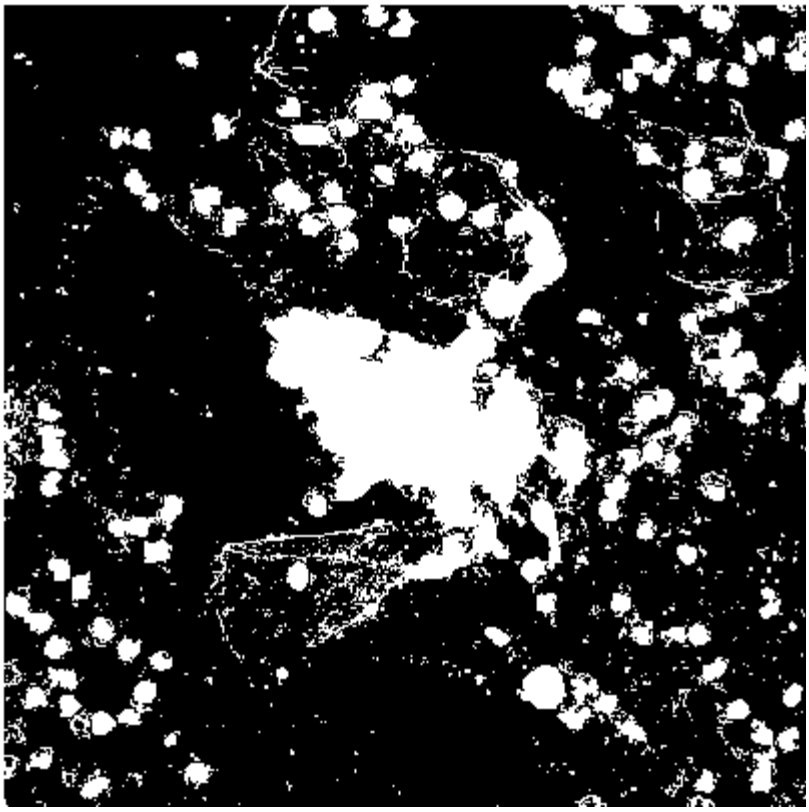


Imagen 9 :





Apéndice B

Publicaciones

Reyes, Andrea et al. “Parallel Design the Algorithm of Digital Images Segmentation Split & Merge”. Memorias Conferencia Latinoamericana de Computación de Alto Rendimiento, Universidad de los Andes, Mérida Venezuela. Pág 81-84, 2009.

Reyes, Andrea et al. “Design of a Parallel Algorithm for Digital Image Segmentation Based on the Split & Merge Method ”. Conferencia Latinoamericana de Computación de Alto Rendimiento, Mérida Venezuela. Acta Científica Venezolana, 60 (3): 118-121, 2009.

Parallel Design the Algorithm of Digital Images Segmentation Split & Merge

Andrea Reyes
Biomedical Engineering Research Group
UIS, Colombia
andrea.reyes@ieee.org

Víctor Martínez
Biomedical Engineering Research Group
UIS, Colombia
viedumar@uis.edu.co

Ana Ramírez
Connectivity and Signal Processing Group
UIS, Colombia
anaberam@uis.edu.co

Abstract

This work presents the design of a parallel algorithm based in the Split & Merge regions method (Horowitz and Pavlidis, 1974) for the Segmentation process of Digital Images, in this case Uterine-Cervix samples.

The proposal utilizes the methodology for the designing parallel algorithm of Ian Foster (Partitioning, Communication, Agglomeration and Mapping).

1. Introduction

Image Segmentation is a key step in Image Processing. Subdivide an image into constituent objects, called regions; it is the process of assigning pixels to each region having common properties, using image attributes such as pixel intensity, spectral values and textural properties [3, 5, 15].

Let R represent the entire image having different objects. An Segmentation of R is a partition of R into subsets $R_1, R_2, \ldots, R_n$, such that [12]:

(a) $U_{i=1}^{n} R_i = R,$

(b) R_i *is a connected region,* $\forall\, i,$

(c) $R_i \cap R_j = \emptyset \;\forall\, i,j, i \neq j,$

(d) $P(R_i) = TRUE \;\forall\, i,$

(e) $P(R_i \cup R_j) = FALSE \;\forall\, i,j, i \neq j,$

Where $P(R_i)$ is a logical predicate over the set of pixels in the set of pixels in R_i and $\emptyset$ is the empty set.

Proposition (a) indicates that segmentation must be complete, that is, every pixel must be in a region while the second one indicates the pixels belonging to a region must be connected. The proposition (c) represents that the regions must be disjointed, and (d) deals with the properties that must be satisfied by the pixels in every segmented region R_i in such a way $P(R_i)$ = TRUE if all pixels in R_i are equivalent with respect R_i and R_j are different in the sense of predicate P.

The Segmentation algorithms can be based in two approaches [5]: First, the discontinuities of the pixels, which the main areas of interest are the detection isolated points and the detection of edges and lines. Second, the similarity of the pixels, where is usual to use algorithms such as threshold or oriented regions (Growth or Split & Merge), since grouped the pixels according to similar properties. The algorithms used for the detection of limits and threshold are not very effective, due to they do not make use of spatial information, so the oriented regions are most suitable for the type of Uterine-Cervix medical images.

The region growing algorithm uses a set of points generators from which it will increases the region, the fundamental problem is to correctly determine the set of initial points. Another alternative is the algorithm Split & Merge, it subdivides the original image into a set of arbitrary disjunct regions and then merge the regions to try to satisfy the criteria of homogeneity, steps that solves the region growing problem. On the other hand, the method based on regions, to determine the location of the final frontier, it has the difficulty of a high consumption of computational resources, which is particularly relevant in applications that require real time response and for this reason, the parallel algorithms constitute a very important approach to solve this problem.

The Split & Merge Algorithm proposed by Horowitz and Pavlidis [2, 4] in 1974 is one of the most popular algorithms for image segmentation.

The algorithm may be summarized by the following steps [3, 8]:

(1) Split any region R_i into four almost equal regions where $P(R_i)$ = FALSE.
(2) Merge any adjacent regions R_i and R_j for which $P(R_i \cup R_j)$ = TRUE.
(3) Stop when no further merging or splitting is possible.
Otherwise repeat steps (1) and (2).

The application area is the Medical Images for this research and the input images are Uterine-Cervix samples. The Cervical Cancer is the second type of cancer most frequently in the female population; the diagnostic is that near 80% of the cases in not developed countries. The test of Cervical Cancer cytology is used to establish the population of women at risk of this cancer, in Bucaramanga (Colombia) finding 1.897 cases of slight displace, 736 moderate displaces, 289 several displaces, in the years 2000 and 2001 [13, 18].

In the Biomedical Engineering Research Group (GIIB) the Industrial of Santander University (UIS), has been performed research works in undergraduates and postgraduates in the field of Digital Image Processing in the detection of Cervical Cancer, however there have been problems in the stage of segmentation, where the main inconvenient to implement an image processing system to support analysis, is that the images acquired in the test are large size and quantity, and the time is quite large [10, 11, 13, 14, 16, 17].

Faced with this situation it is proposed to design a parallel algorithm based on the regions method Split & Merge, with the purposed to posterior implement and evaluate the reduction in the execution time, since at present has not been achieved an optimal sequential implementation, due to that in different tests performed has been come to use all the computational resources available without to obtain a result.

2. Split & Merge Regions Method

The Algorithm Split & Merge requires two types of operations [3]; a fast split phase which is followed by one or more merge phases. The split stage partitions an image into square regions which conform to a first homogeneity criterion; then merge these square regions into larger regions which conform to a second homogeneity criterion.

The Split stage consists of generating a quad-tree (see Fig. 1), where nodes at level i are the result of splitting the sub- images of size $N_1 / 2^i$ x $N_2 / 2^i$ into four sub-images of size $N_1 / 2^{(i+1)}$ x $N_2 / 2^{(i+1)}$ [3, 5]. The construction of this quad-tree can be carried out following a Bottom-up strategy of merging sub-images or Top-Down strategy of splitting [2, 4].

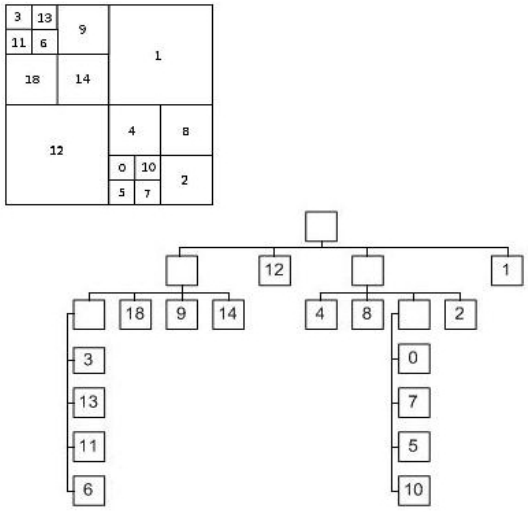


Figure 1. Quad-tree representation

In both strategies, only those nodes satisfying the homogeneity criterion will be generated. The Split phase ends when no more square regions can be generated. The set of leaves (regions) on the quad-tree produced by the Split phase will be the set of input regions for the Merge phase.

The homogeneity criterion of the one region is given by the gray levels of pixels that comprise, establishing a criterion of uniformity. Let $X_{i,\ i=1,2,3,...n}$ the gray levels of pixels that make up a region R. The average value of gray levels in the region R, is the arithmetic average of gray levels of pixels [9]:

$$\bar{X}_R = \frac{\sum_{i=1}^{n} X_i}{n} \qquad (1)$$

A region R is uniform gray levels, if for every sub-regions $r_i \in R$, is satisfied that the ratio between the minimum of the average values of gray levels of the R region and a sub-region r_i and the maximum of these values is less than or equal to 0.95. That is,

$$\frac{Min\left(\bar{X}_R, \bar{X}_{ri}\right)}{Max\left(\bar{X}_R, \bar{X}_{ri}\right)} \leq 0{,}95 \ \forall\, ri, i = 1, \ldots 4 \qquad (2)$$

And the region is homogeneity in gray levels if it satisfies with the criterion of uniformity.

3. Design Parallel Algorithm of Ian Foster

This methodology structures the design process as four following stages (see Fig. 2). In the first and two stages, the focus is on concurrency and scalability. In the third and fourth stages, attention shifts to locality and performance [1].

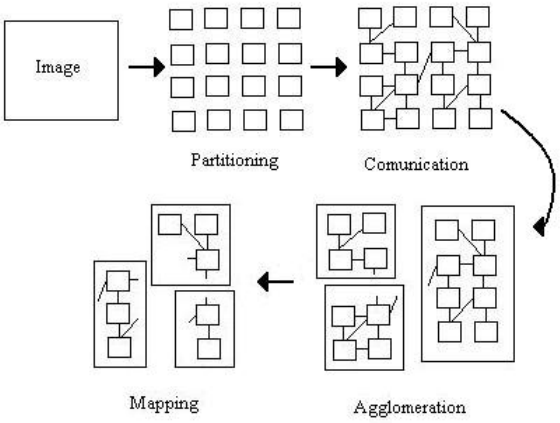


Figure 2. Methodology for design parallel programs

i) Partitioning: The partitioning stage of a design expose opportunities for parallelism. The focus is on defining a large number of small tasks, fine-grained decomposition of a problem.

ii) Communication: The tasks require data associated with another task. Data must then be transferred between tasks so as to allow computation to proceed. In the communication is defining a channel structure that links and specifies the messages that are to be sent and received on these channels. The communication can be classified in local-global, structured-unstructured, static-dynamic, and synchronous-asynchronous.

iii) Agglomeration: The task and communication structures determined in the two previous stages are evaluated against the requirements of performance and costs implementation, to obtaining an algorithm that will execute efficiently on some class of parallel computer. If necessary, the tasks are combined to increase performance and reduce the communication and may still be greater than the number of processors.

iv) Mapping: In the final stage, specify where each task is to execute and assigned to each processor. Operating system or hardware mechanisms can be relied upon to schedule executable tasks to available processors. The goal in developing mapping algorithms is minimize total execution time.

4. Proposed Algorithm

i) Partitioning: The Split step is inherently parallel task. Initially, an image of size $N = N_1 \times N_2$ is partitioned into $P = P_1 \times P_2$ sub-images that containing adjacent pixels $N_1/P_1 \times N_2/P_2$ [6]. The structure for represent the regions of the image is the Quad-tree [5, 7].

ii) Communication: Each processor work splitting and merging regions, this corresponds to either removing or building nodes of the tree structure [5], communicates transferred data of the adjacency node, this information is stored using an adjacency matrix. Each processor communicates too with others processors for merge regions, satisfying the homogeneity criterion, and the constraints (a)-(e) in the steps (1)-(3) (see section 1).

iii) Agglomeration: In the quad-tree representation are defined the method for identifying regions (Id), for this method use a cyclic and random strategy [3, 6], each processor assigns an Id to each region. The Ids should be different for each region and regions in every processor. The Id assigned in the quad-tree contains the information the vertices and edge the region. For select the task concurrently use a load balancing describes in the next step.

iv) Mapping: The process of the merge between regions are data dependent and the performance of communication, agglomerate and map the algorithm decreasing with the execution time, in this case the algorithms for static and dynamic load balancing are necessaries. For the load balance locally the dynamic balanced algorithm, assigning identifiers cyclic, is suitable and the dynamic balanced algorithm with assignment identifiers random for the load balance global [3].

The parallel algorithm can be describes by the following steps [3, 6, 5, 7, 8]:

Step I: Image N is partitioned in P1 x P2 sub-images and mapped onto the set of Processors.
Step II: Each Processor realized the Split on its own sub-image and assigns an identifier (Id) to each of the created regions.
Step III: Each Processor builds its own local tree by creating the set of edges between its own local regions which satisfy homogeneity criteria.
Step IV: Processors exchange information about the regions at the border of the initial sub-image and create the set of edges between regions that belong to different neighboring Processor.
Step V: Each region determines the linked regions that best suit the homogeneity criterion as regions to be merged. If more than one region are candidates to

merge with it will be selected that region with the minimum Id.
Step VI: Processors must exchange information about the best selection for those linked regions located at different Processor.
Step VII: The vertices and edges of the tree are updated for the current set of regions.
Step VIII: Run the algorithm for load balancing.
Step IX: Repeat steps V–VIII while linked regions still exist.
Otherwise the algorithm ends.

5. Discussions and Future Work

The algorithm proposed is for implement in a SPMD programming model and working in architecture MIMD with the message passing MPI.

The importance of this research is given to contribute to the development of the line of research on Digital Image Processing in the area of High Performance Computing on the GIIB; in the solution to the problem of the cost of time processing images Uterine-Cervix for diagnostic Cervical Cancer.

6. References

[1] Foster, I. "Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering". *Addison-Wesley Longman Publishing Co*., Inc. 1995.

[2] Horowitz, S.L. and Pavlidis, T. "Picture segmentation by a directed split-and-merge procedure". *Proceedings, 2nd International J. C. on Pattern Recognition, Copenhagen,* 424–433. 1974

[3] Montoya, M. D. Gil, C. and García, I. "The load unbalancing problem for region growing image segmentation algorithms". *J. Parallel Distrib. Comput.* 63, 4 (Apr. 2003), 387-395. 2003.

[4] Horowitz, S. L. and Pavlidis, T. "Picture Segmentation by a Tree Traversal Algorithm". *J. ACM 23, 2 (Apr. 1976*), 368-388. 1976.

[5] Kelkar, D. and Gupta, S. "Improved Quadtree Method for Split Merge Image Segmentation". I*n: Proceedings, First international Conference on Emerging Trends in Engineering and Technology - Volume 00 (July 16 - 18, 2008). ICETET. IEEE Computer Society, Washington, DC,* 44-47. 2008.

[6] Montoya, M.D.G.; Gil, C.; Garcia, I., "Load balancing for a class of irregular and dynamic problems: region growing image segmentation algorithms," *Parallel and Distributed Processing. Proceedings of the Seventh Euromicro Workshop on* , vol., no., pp.163-169, 3-5, Feb 1999.

[7] Tyagi, A.; Bayoumi, M., "Systolic array implementation of image segmentation by a directed split and merge procedure," *Pattern Recognition, 1990. Proceedings., 10th International Conference on* , vol.ii, no., pp.491-493 vol.2, 16-21, Jun 1990.

[8] Faruquzzaman, A.B.M.; Paiker, N.R.; Arafat, J.; Karim, Z.; Ameer Ali, M., "Object segmentation based on split and merge algorithm," *TENCON 2008 - 2008. IEEE Region 10 Conference* , vol., no., pp.1-4, 19-21, Nov. 2008.

[9] Alvarez M, Rivas M. and Rukoz M. "Segmentación de Imágenes Biomédicas Mediante el Crecimiento de Regiones". *Acta Científica Venezolana,* 52: 192–198, 2001

[10] Martinez, V. et al. 2007. "Tecnología Grid para la Detección de Cáncer de Mama y Cuello Uterino por Medio de Procesamiento de Imágenes". *CLCAR 2007. Colombia, 13-18 Agosto* 2007, 318-324.

[11] Niño, D. et al. "Contribución al estudio de las células escamosas de citologías cérvico uterinas que presentan cambio por ASCUS, por medio de tratamiento digital de imágenes". *Trabajo de Grado. UIS. 2006.*

[12] Gonzalez, R.C. Woods, R.E. "Digital Image Processing", *2nd Edition, Prentice-Hall, Inc., Upper Saddle River, NJ*. 2002.

[13] Martinez, V. et al. "Computational model for squamous cells characterization during cervical smear cytology" . *Rev. Colomb. Biotecnol. Vol. VII No. 2 Diciembre* 2005, 35-46. 2005.

[14] Amaya, M.L. et al. "Contribución al estudio de las Células Escamosas de Citologías Cérvico Uterinas que Presentan Cambios por Virus de Papilona Humano (VPH), utilizando Tratamiento Digital de Imágenes". *Trabajo de Grado. UIS.* 2006.

[15] Tilton, J.C., "Image segmentation by iterative parallel region growing with applications to data compression and image analysis," *Frontiers of Massively Parallel Computation. Proceedings., 2nd Symposium on the Frontiers of* , vol., no., pp.357-360, 10-12 Oct 1988

[16] Martinez, V. "Modelo Computacional para Caracterización de células endocervicales". *Trabajo de Maestria. UI*S. 2007.

[17] Martinez, V.E. et al. "Software Tool for Squamous Cells Classification in Cervical Smear Cytologies" *ICCB2005: Proceedings of II International Conference on Computational Bioengineering, IST Press, Lisbon*, 2:1089-1094. 2005.

[18] Universidad Autónoma de Bucaramanga. 2002. "Registro de población de cáncer en el área metropolitana de Bucaramanga". 2000-2001.

Conferencia Latinoamericana de Computación de Alto Rendimiento
Acta Científica Venezolana, 60 (3): 118-121, 2009

DESIGN OF A PARALLEL ALGORITHM FOR DIGITAL IMAGE SEGMENTATION BASED ON THE SPLIT & MERGE METHOD

Reyes, Andrea; Martínez, Víctor; Ramírez, Ana

Grupo de Investigación en Ingeniería Biomédica, Universidad Industrial de Santander, Bucaramanga, Colombia

Abstract. We present the design of a parallel algorithm based on the Split & Merge regions method (Horowitz and Pavlidis, 1974) for the segmentation of digital images, in our case uterine-cervix samples. The proposal makes use of the methodology for designing parallel algorithms by Ian Foster, namely partitioning, communication, agglomeration and mapping. **Key words:**

DISEÑO DE UN ALGORITMO PARALELO PARA LA SEGMENTACIÓN DE IMÁGENES DIGITALES BASADO EN EL MÉTODO DE SPLIT & MERGE

Resumen. Presentamos el diseño de un algoritmo paralelo basado en el método de Split & Merge (Horowitz and Pavlidis 1974) para la segmentación de imagines digitales, en nuestro caso muestras cérvico-uterinas. La propuesta hace uso de la metodología de Ian Foster para diseñar algoritmos paralelos, específicamente partición, comunicación, aglomeración y mapeo. **Palabras claves:**

INTRODUCTION

Image segmentation is a key step in image processing whereby an image is subdivided in constituent objects called *regions*, pixels being assigned to regions having common properties using image attributes such as pixel intensity, spectral values and textural properties [8,13,16]. Let R represent the entire image having different objects; the segmentation of R is a partition of R in the subsets R_1, R_2,..., R_n, such that [5]

$$U_{i=1}^{n} R_i = R, \quad (1)$$

$$R_i \text{ is a connected region}, \forall i, \quad (2)$$

$$R_i \cap R_j = \varnothing \forall i, j, i \neq j, \quad (3)$$

$$P(R_i) = TRUE \forall i, \quad (4)$$

$$P(R_i \cup R_j) = FALSE \forall i, j, i \neq j, \quad (5)$$

where $P(R_i)$ is a logical predicate over the set of pixels in R_i and $\varnothing$ is the empty set. Proposition (1) indicates that segmentation must be complete, that is, every pixel must be in a region while (2) specifies that pixels belonging to a region must be connected. Proposition (3) prescribes that the regions must be disjointed, and (4) deals with the properties that must be satisfied by the pixels in every segmented region R_i such that $P(R_i)$ = TRUE if all pixels in R_i which are equivalent with respect to R_i and R_j are different in the sense of the predicate P.

Segmentation algorithms can be based on two approaches [8]. Firstly, pixel discontinuities in which features of interest are the detection of isolated points and edges and lines. Secondly, pixel similarity where algorithms such as threshold or oriented regions (Growth or Split & Merge) are frequently used to group pixels according to similar properties. The algorithms used in the detection of limits and thresholds are not very effective since they do not make use of spatial information; thus the oriented regions are the most suitable for uterine-cervix type of medical images.

The region growing algorithm uses point generators from which the region will be increased. The fundamental problem is to correctly determine the set of initial points. An alternative is the Split & Merge algorithm where the original image is subdivided into a set of arbitrary disjoint regions which are then merged complying with criteria of homogeneity, steps that resolve the region growing problem. On the other hand, to determine the location of the final frontier, the method based on regions encounters the difficulty of a high consumption of computational resources, which is particularly relevant in applications that require real-time response. For this reason, the parallel algorithms become a relevant approach to solve this problem.

The Split & Merge Algorithm proposed by Horowitz and Pavlidis [6,7] in 1974 is one of the most popular for image segmentation. A summary of the algorithm is given by the following steps [3,13].

1. Any region R_i is split into four almost equal sub-regions where $P(R_i)$ = FALSE;
2. Any adjacent regions R_i and R_j for which P(R_i U R_j) = TRUE are merged;
3. It stops when no further merging or splitting is possible, otherwise steps 1 and 2 are repeated.

The application area for the present work is medical images, input images being uterine-cervix samples. Cervical cancer is the second most frequent type of cancer in the female population, nearly 80% of the cases in underdeveloped countries. Cervical-cancer cytology tests are used to establish the population of women at risk; for instance, in 2000 and 2001, 1.897 cases of were found in Bucaramanga, Colombia, with slight displaces, 736 with moderate displaces and 289 with severe displaces [11,18].

The Biomedical Engineering Research Group (GIIB) at the Universidad Industrial de Santander (UIS) is undertaking research in the field of digital image processing to detect cervical cancer, involving both undergraduate and postgraduate students. However, problems have been encountered at the segmentation stage, where the main inconvenience in the implementation of an image processing system to support analysis is test-

image size and quantity, processing time being quite large as well [2, 9, 10,11,12,15]. To improve this situation, we propose the design of a parallel algorithm based on the regions Split & Merge method such that it can be subsequently implemented to evaluate the execution-time reduction. Moreover, an optimal sequential implementation has not been achieved since during the tests performed all the available computational resources were used without obtaining a result.

SPLIT & MERGE REGIONS METHOD

The Split & Merge algorithm requires two types of operations [13]: a fast-split phase followed by one or more merge phases. In the split stage, an image is partitioned into square regions which conform to a first homogeneity criterion; these square regions are then merged into larger regions which conform to a second homogeneity criterion. The split stage consists in generating a quad-tree (see Fig. 1), where nodes at level i are the result of splitting the sub-images of size $N_1 / 2^i$ x $N_2 / 2^i$ into four sub-images of size $N_1 / 2^{(i+1)}$ x $N_2 / 2^{(i+1)}$ [8,13]. The construction of this quad-tree can be carried out following a bottom-up strategy to merge sub-images or a splitting top-down strategy [6,7]. In both strategies, only those nodes satisfying the homogeneity criterion are generated. The split phase ends when no more square regions can be generated. The set of leaves (regions) in the quad-tree produced by the split phase are the set of input regions for the merge phase.

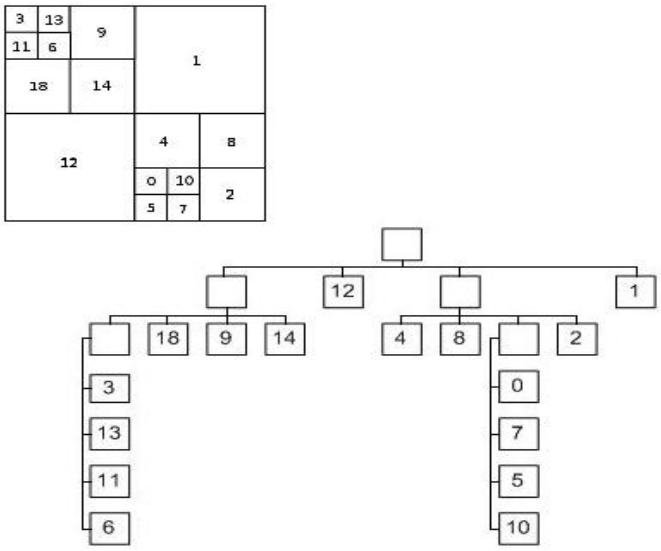


Figure 1. Quad-tree representation.

The homogeneity criterion for a region is given by the gray-level pixels that obey a criterion of uniformity.

Let $X_{i,\ i=1,2,3,...n}$ be the gray-level pixels that make up a region R. The gray-level average value in region R is the arithmetic average of gray-level pixels [1]

$$\overline{X}_R = \frac{\sum_{i=1}^{n} X_i}{n} \qquad (6)$$

A region R is uniform in gray levels if, for every sub-regions $r_i \in R$, the ratio between the minimum of the average values of gray levels of region R and a sub-region r_i is satisfied, and the maximum of these values is less than or equal to 0.95; that is,

$$\frac{Min\left(\overline{X}_R, \overline{X}_{ri}\right)}{Max\left(\overline{X}_R, \overline{X}_{ri}\right)} \leq 0{,}95 \forall ri, i = 1,...4 \qquad (7)$$

and the region is homogeneous in gray levels if it satisfies the criterion of uniformity.

PARALLEL ALGORITHM DESIGN

The methodology by Ian Foster structures the design process in four sequential stages (see Fig. 2). In the first and second stages, the focus is on concurrency and scalability while in the third and fourth stages attention shifts to locality and performance [4].

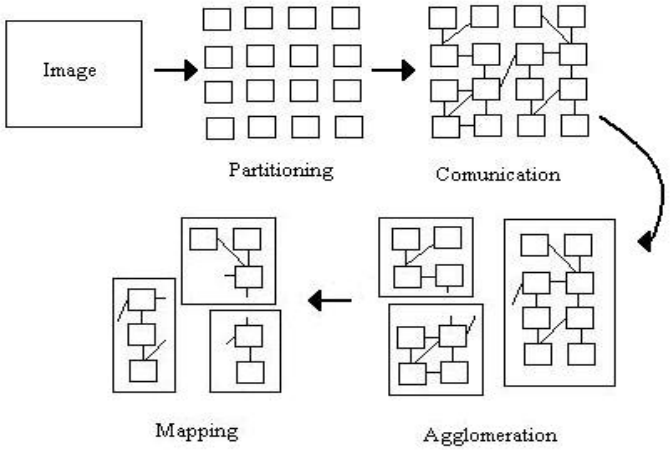


Figure 2. Methodology for designing parallel programs.

1. **Partitioning.** The partitioning stage of a design exposes opportunities for parallelism. The focus is on defining a large number of small tasks, that is, *fine-grained* decomposition of a problem.
2. **Communication.** The tasks require data associated with another task. Data must then be transferred between tasks so as to allow the computation to proceed. The communication defines a channel structure that links and specifies the messages that are to be sent and received on these channels. The communication can be classified in: local-global, structured-unstructured, static-dynamic and synchronous-asynchronous.
3. **Agglomeration.** The task and communication structures determined in the two previous stages are evaluated against the requirements of performance and costs implementation, obtaining an algorithm that will execute efficiently on some class of parallel computers. If necessary, the tasks are combined to increase performance and reduce communication, and may still be greater than the number of processors.
4. **Mapping.** The final stage specifies where each task is to execute and assigned to each processor.

Operating system or hardware mechanisms can be relied upon to schedule executable tasks to available processors. The goal in developing mapping algorithms is to minimize the total execution time.

PROPOSED ALGORITHM

1. **Partitioning.** The split step is inherently a parallel task. Initially, an image of size $N = N_1 \times N_2$ is partitioned into $P = P_1 \times P_2$ sub-images containing adjacent pixels $N_1/P_1 \times N_2/P_2$ [14]. The structure for representing the image regions is the quad-tree[8,17].
2. **Communication**. Each processor works splitting and merging regions which corresponds to either removing or building nodes of the tree structure [8]. Transferred data are communicated to the adjacent node where this information is stored using an adjacency matrix. Each processor communicates as well with others processors which merge regions, satisfying the homogeneity criterion and the constraints (1)-(5) specified in the Introduction.
3. **Agglomeration.** In the quad-tree representation, the method for identifying regions (Id) is defined which uses a cyclic and random strategy [13,14], each processor assigning an Id to each region. Ids should be different for each region and regions in every processor. The assigned Id in the quad-tree contains the information of the region vertices and edge. Task concurrent selection uses a load balancing described in the next step.
4. **Mapping.** Since the merge processes between regions are data dependent and the algorithm performance for communicating, agglomerating and mapping decreases with execution time, algorithms for static and dynamic load balancing are necessary. For local load balancing, the dynamic balanced algorithm with cyclic identifier assignment is suitable, while the dynamic balanced algorithm with random identifier assignment can be used for global load balancing[13].

The parallel algorithm can be described by the following steps [3,8,13,14,17].

1. Image P1 x P2 is partitioned in four sub-images and mapped onto the set of processors.
2. Each processor performs the split on its own sub-image and assigns an identifier (Id) to each of the created regions.
3. Each processor builds its own local tree by creating the set of edges between its own local regions which satisfy the homogeneity criteria.
4. Processors exchange information about the border regions of the initial sub-image, and create the set of edges between regions that belong to different neighboring processors.
5. Each region determines the linked regions that suit best the homogeneity criterion as regions to merge. If more than one region are candidates to merge, the region with the minimum Id will be selected.
6. Processors must exchange information about the best selection for those linked regions located at a different processor.
7. The tree vertices and edges are updated for the current set of regions.
8. The algorithm is run for load balancing.
9. Repeat steps (5)-(8) while linked regions still exist, otherwise the algorithm ends.

DISCUSSION AND FUTURE WORK

The proposed algorithm is for implementing an SPMD programming model on an MIMD architecture with MPI message passing. The importance of this work is its contribution to the development of HPC digital image processing at the GIIB, particularly in the reduction of execution time when dealing with uterine-cervix images for the diagnostic of cervical cancer.

REFERENCES

1. **Alvarez, M., Rivas, M., Rukoz, M.** Segmentación de imágenes biomédicas mediante el crecimiento de regiones. *Acta Científica Venezolana,* **52**:192-198, 2001.
2. **Amaya, M.L., et al.** Contribución al estudio de las células escamosas de citologías cérvico uterinas que presentan cambios por virus de papilona humano (VPH), utilizando tratamiento digital de imágenes. Trabajo de Grado, Universidad Industrial de Santander, 2006.
3. **Faruquzzaman, A.B.M., Paiker, N.R., Arafat, J., Karim, Z., Ameer A. M.** Object segmentation based on split and merge algorithm. En *IEEE Region 10th Conference*, 2008, pp.1-4.
4. **Foster, I.** Designing and Building Parallel Programs: concepts and tools for parallel software engineering. Addison-Wesley, 1995.
5. **Gonzalez, R.C., Woods, R.E.** Digital Image Processing, 2nd Edition, Prentice-Hall, Upper Saddle River, NJ, 2002.
6. **Horowitz, S.L., Pavlidis, T.** Picture segmentation by a directed split-and-merge procedure. En *Proceedings, 2nd International J. C. on Pattern Recognition, Copenhagen,* 1974, pp. 424-433.
7. **Horowitz, S.L., Pavlidis, T.** Picture segmentation by a tree traversal algorithm. *JACM,* **23:**368-388, 1976.
8. **Kelkar, D., Gupta, S.** Improved quad-tree method for split merge image segmentation. En *Proceedings, First international Conference on Emerging Trends in Engineering and Technology (ICETET),* IEEE Computer Society, Washington, DC, 2008, pp. 44-47.
9. **Martínez, V.** Modelo computacional para caracterización de células endocervicales. Trabajo de Maestría, Universidad Industrial de Santander, 2007.
10. **Martínez, V., et al.** Tecnología grid para la detección de cáncer de mama y cuello uterino por medio del procesamiento de imágenes. En Conferencia Latinoamericana de Computación de Alto Rendimiento 2007, C.J. Barrios-Hernández (ed), Editorial SIC, Bucaramanga, 2007, pp. 318-324.

11. **Martinez, V., et al.** Computational model for squamous cells characterization during cervical smear cytology. *Rev. Colomb. Biotecnol.,* **3**:35-46, 2005.
12. **Martínez, V.E., et al.** Software tool for squamous cells classification in cervical smear cytologies. En *Proceedings of II International Conference on Computational Bioengineering, IST Press, Lisbon*, 2005, pp. 1089-1094.
13. **Montoya, M.D., Gil, C., García, I.** The load unbalancing problem for region growing image segmentation algorithms. *J. Par. Distrib. Comput.,* **63**:387-395, 2003.
14. **Montoya, M.D.G., Gil, C., Garcia, I.** Load balancing for a class of irregular and dynamic problems: region growing image segmentation algorithms. En *Proceedings of the Seventh Euromicro Workshop on Parallel and Distributed Processing*, 1999, pp.163-169.
15. **Niño, D., et al.** Contribución al estudio de las células escamosas de citologías cérvico uterinas que presentan cambio por ASCUS, por medio de tratamiento digital de imágenes. Trabajo de Grado, Universidad Industrial de Santander, Bucaramanga, 2006.
16. **Tilton, J.C.** Image segmentation by iterative parallel region growing with applications to data compression and image analysis. En *Proceedings, 2nd Symposium on the Frontiers of Massively Parallel Computation*, 1988, pp. 357-360.
17. **Tyagi, A., Bayoumi, M.** Systolic array implementation of image segmentation by a directed split and merge procedure. En *Proceedings, 10th International Conference on Pattern Recognition*, vol. 2, 1990, pp.491-493.
18. **Universidad Autónoma de Bucaramanga**. Registro de población de cáncer en el área metropolitana de Bucaramanga. 2002.

Correspondencia: Andrea Reyes, Grupo de Investigación en Ingeniería Biomédica, Universidad Industrial de Santander, Bucaramanga, Colombia.

Correo electrónico: *andrea.reyes@ieee.org*