**UNIVERSIDAD DE SANTIAGO DE CHILE**
**FACULTAD DE INGENIERÍA**
**DEPARTAMENTO DE INGENIERÍA INFORMÁTICA**


## Sistemas Operativos

# Informe Laboratorio Nº3

**Jorge Paredes**
**Sebastián Pinto-Agüero**

Profesor:
Cristóbal Acosta

Ayudantes:
Nestor Mora
Marcela Rivera

Santiago - Chile
2017

# TABLA DE CONTENIDOS



# ÍNDICE DE FIGURAS

# 1. INTRODUCCIÓN

## 1.1 CONTEXTO

Uno de los temas principales dentro del curso es el estudio de las hebras y cómo implementarlas, de tal forma que se una hebra no interfiera con otra y el resultado de las operaciones no se vea afectado. De ser necesario, es importante establecer un mecanismo de sincronización y así evitar tener problemas al término de la ejecucón del programa.

## 1.2 OBJETIVO

El objetivo de este laboratorio es implementar un simulador paralelo de la difusión de una onda según la equación de Schroendinger. Para esto, se pide utilizar Pthread como tecnología de paralelización.

## 1.3 ECUACIÓN A IMPLEMENTAR

La ecuación de Schroendinger para una grilla de NxN se define como:

$$H_{i,j}^{t} = 2H_{i,j}^{t-1} - H_{i,j}^{t-2} + c^2\left(\frac{at}{dd}\right)^2\left(H_{i+1,j}^{t-1} + H_{i-1,j}^{t-1} + H_{i,j-1}^{t-1} + H_{i,j+1}^{t-1} - 4H_{i,j}^{t-1}\right)$$

Donde **H** es la matriz que contiene la altura de la onda en cada posición de la grilla en el tiempo **t**, **c** es la velocidad de la onda en el medio, **dt** es el intervalo de tiempo con que avanza la simulación y **dd** es el cambio en la superficie. Para este laboratorio considerar **c** = 1.0, **dt** = 0.1, **dd** = 2.0.

La condición de borde es la siguiente:

$$H_{0,j}^{t} = H_{i,0}^{t} = H_{N-1,j}^{t} = H_{i,N-1}^{t} = 0 \quad \forall i, j, t$$

La condición inicial:

$$H_{i,j}^{0} = 20 \quad \forall\, 0.4N < i < 0.6N,\ \ 0.4N < j < 0.6N$$

La ecuación para la primera iteración:

$$H_{i,j}^{t} = H_{i,j}^{t-1} + \frac{c^2}{2}\left(\frac{dt}{dd}\right)^2\left(H_{i+1,j}^{t-1} + H_{i-1,j}^{t-1} + H_{i,j-1}^{t-1} + H_{i,j+1}^{t-1} - 4H_{i,j}^{t-1}\right)$$

# 2. GRÁFICOS Y RESULTADOS

Para calcular el tiempo de ejecución del programa, se ha hecho uso del programa *time* en un computador del laboratorio y se ha considerado los valores correspondientes al tiempo real. Los gráficos han sido realizados en Matlab. En la esquina superior derecha se muestra a cuantas iteraciones corresponden los puntos del gráfico.

## 2.1 TIEMPO DE EJECUCIÓN

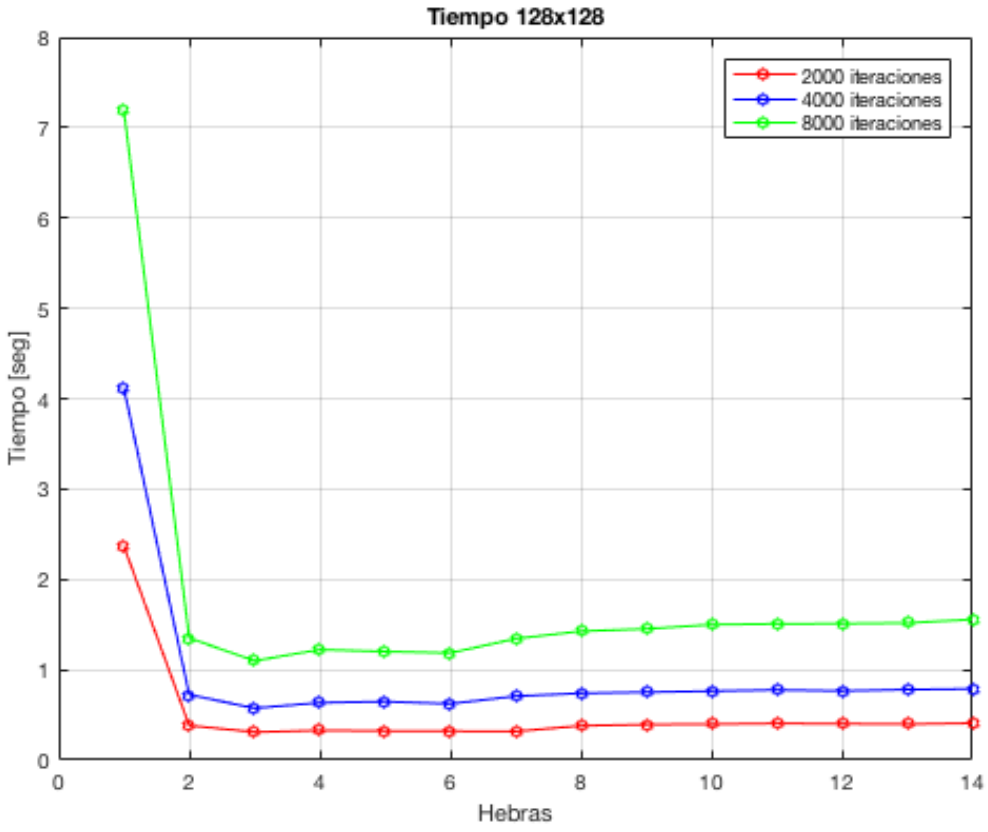


**Figura 1: Tiempo de ejecución con una grilla de 128x128**

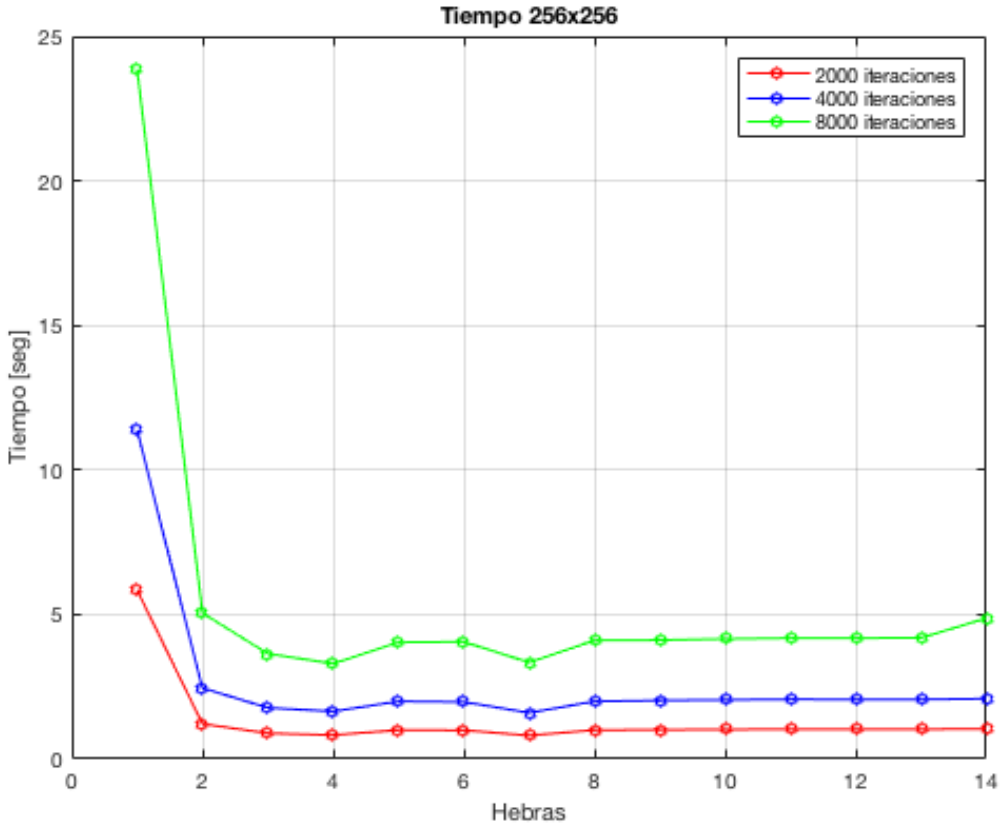


**Figura 2: Tiempo de ejecución para una grilla de 256x256**

## 2.2 SPEEDUP

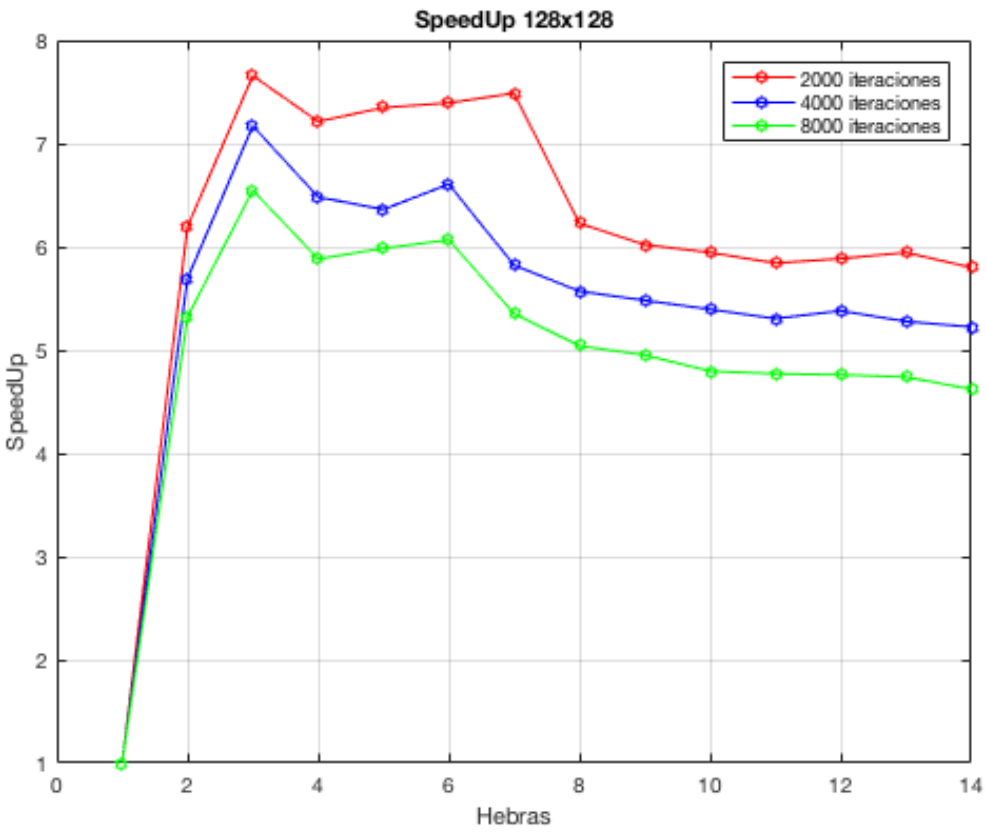


**Figura 3: SpeedUp para una grilla de 128x128**

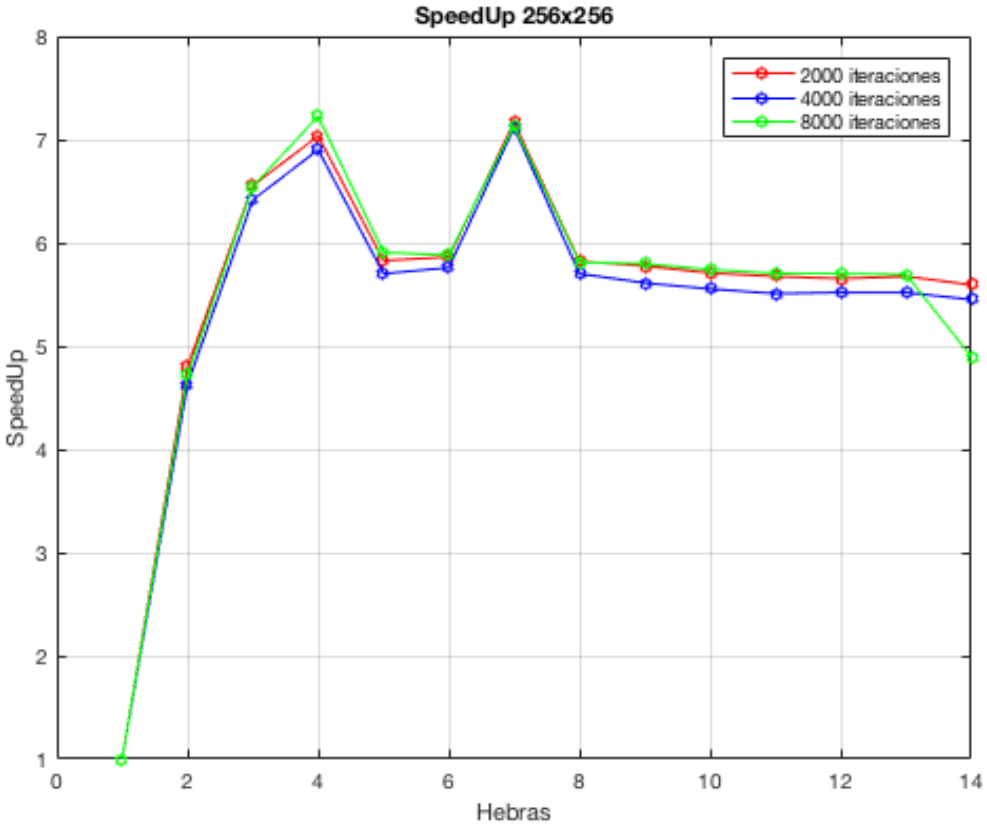


**Figura 4: SpeedUp para una grilla de 256x256**

## 2.3 EFICIENCIA

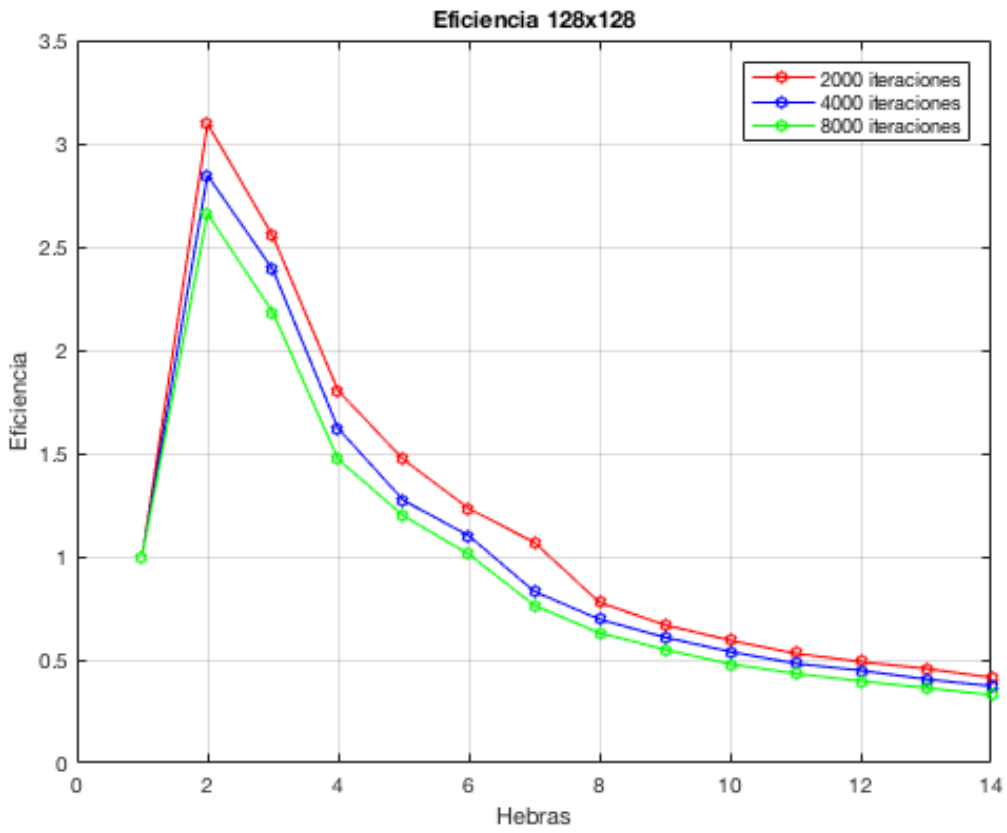


**Figura 5: Eficiencia para una grilla de 128x128**

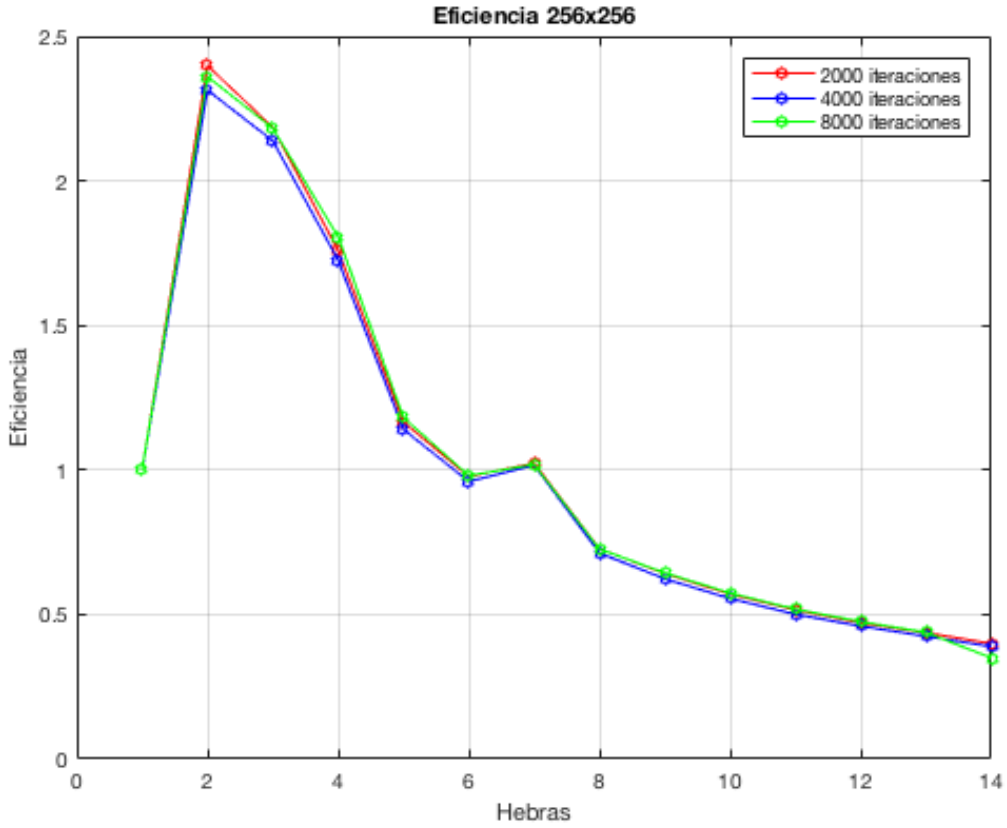


**Figura 6: Eficiencia para una grilla de 256x256**

# 3. ESTRATEGIA DE PARALELIZACIÓN

En esta implementación, a cada una de las hebras creadas se les asigna un área específica de la grilla, de tal forma que en una sección de la grilla se encuentre una sola hebra realizando la ecuación de Schroendinger. Como todas las hebras tienen su sección definida, nunca interferirán con el trabajo de otra, por lo que no es necesario implementar un método de sincronización entre hebras (por ejemplo, un *mutex*). Esto es porque cada una de las secciones de la grilla corresponde a un recurso que es asignado a una hebra en específico, la cual sólo se encarga de acceder a esa sección y de esta forma se asegura exclusión mútua.

El programa utiliza un arreglo de tres dimensiones (un cubo) para realizar las iteraciones en la grilla, siendo la grilla dos dimensiones y las iteraciones la tercera. Es decir, las dimensiones del cubo son NxNxT. A medida que se avanza en las iteraciones, las hebras de una determinada sección utilizan los resultados obtenidos anteriormente en la misma sección, ya que nunca cambia en el espacio de dos dimensiones. Una sección anterior es accedida por una única hebra y no modifica los resultados, por lo que se asegura exclusión mútua.

# 4. ANÁLISIS DE RESULTADOS Y CONCLUSIÓN

El tiempo de ejecución para una grilla de 128x128 más alto encotrado ocurre con 1 sola hebra, mientras que el menor tiempo es con 3 hebras para 2000, 4000 y 8000 iteraciones. Después de eso, mientras más hebras se encuentran trabajando el tiempo de ejecución tiende a aumentar lentamente. Para el caso de la grilla de 256x256, el tiempo de ejecución es mayor con 1 hebra, luego el tiempo baja drásticamente con 2 hebras y alcanza un valor mínimo con 7 hebras. Finalmente, el tiempo vuelve a aumentar levemente a medida que se incrementa el número de hebras. Esto es válido para 2000, 4000 y 8000 iteraciones.

El speed up es calculado de acuerdo al tiempo de ejecución que tarda el programa para 1 a 14 hebras utilizando la siguiente fórmula:

$$S(n) = T(1)/T(n)$$

Donde **n** son la cantidad de hebras utilizadas, **S(n)** es el speed up obtenido con **n** hebras y **T(n)** el tiempo que tarda el programa en ejecutarse con **n** hebras.

Para una grilla de 128x128, el mayor speed up se alcanza con 3 hebras y luego tiene a bajar léntamente hasta llegar a 14 hebras, con excepción cuando hay 6 hebras donde el speed up sube nuevamente; lo anterior es válido para 2000, 4000 y 8000 iteraciones. Con una grilla de 256x256 alcanza un speed up alto con 4 hebras y luego baja, hasta llegar a 7 hebras donde alcanza un valor alto y nuevamente baja, y a medida que se aumentan las hebras hasta llegar a 14 el speed up disminuye léntamente. Con 2000 y 4000 iteraciones se alcanza mayor speed up con 7 hebras y para 8000 iteraciones el speed up es mayor con 4 hebras.

La eficiencia se calcula de acuerdo al speed up alcanzado para 1 a 14 hebras y al número de hebras utilizado. Se utiliza la siguiente fórmula:

$$E(n) = \frac{S(n)}{n} = \frac{T(1)}{n * T(n)}$$

Donde **n** son el número de hebras utilizadas, **S(n)** es el speed up para **n** hebras y **E(n)** es la eficiencia para n hebras.

Para una grilla de 128x128 y de 256x256 la mayor eficiencia se alcanza con dos hebras y luego tiene a disminuir a medida que se incrementa la cantidad de hebras. En la grilla de 256x256 hay un aumento de eficiencia con 7 hebras respecto al valor obtenido con 6 hebras, pero luego sigue disminuyendo hasta llegar a 14 hebras. Esto es válido para 2000, 4000 y 8000 iteraciones.

Se puede apreciar que no siempre el tiempo de ejecución, speed up y eficiencia de un programa aumentarán mientras más hebras se utilicen, ya que llegará un punto en el cual al utilizar más hebras no ocurrirán mejoras e incluso podrían perjudicar la ejecución del programa, tal como se puede observar en los gráficos. Llega un punto en el cual el tiempo, speed up y eficiencia es mejor, pero luego al incrementar el número de hebras los resultados comienzan a empeorar. Estos resultados no solo dependen del programa implementado o estrategia de paraelización, sino también de las características del equipo y el sistema operativo en sí, quien se encarga de planificar el orden de ejecución de los procesos y hebras. Esta ejecucuión puede verse afectada por otros procesos que están siendo ejecutados en ese instante, lo cual puede causar variaciones en el tiempo de ejecución.

El objetivo de este laboratorio se ha cumplido sin mayor dificultad. Se puede ver que los resultados experimentales cumplen con la teoría, pero en algunos casos hay algunas leves variaciones respecto a la teoría, esto es debido a las características del equipo en el cual se está ejecutando el programa y que al igual que como ocurre en física o en otras ciencias, los resultados experimentales siempre difieren de la teoría a causa de factores externos que afectan a los resultados (en este caso, pueden ser otros procesos que están siendo ejecutados) mientras que la teoría considera condiciones ideales.

# 5. REFERENCIAS

William Stallings. Sistemas Operativos: Aspectos internos y principios de diseño. Quinta edición.