Trabajo práctico 0: Ley de Amdahl

1. Introducción

En la clase del 22/8 se presentaron las herramientas necesarias para compilar, depurar y ejecutar programas en un ambiente MIPS. Asímismo se describieron los fundamentos de la Ley de Amdahl[1], y como puede ser usada para estimar y medir los alcances de una mejora a un proceso.

Este trabajo práctico consiste en estudiar la performance de una implementación de la Hormiga de Langton que denominaremos 'La hormiga artista'.

2. La hormiga artista

Como parte de los recursos a utilizar en este trabajo práctico, se provee un número de implementaciones del problema. Para todos los casos, se deben especificar los siguientes parámetros

- Las dimensiones de la grilla
- La paleta de colores a utilizar
- Un conjunto de reglas para realizar las rotaciones
- La cantidad de movimientos a realizar

A su vez, se pueden especificar las siguientes opciones de compilación para generar distintas versiones

- `USE_COL_MAJOR` Al momento de imprimir el estado de la grilla, la misma es recorrida 'por columnas'
- `USE_TABLES` Las acciones de rotar y moverse hacia adelante se realizan en funciones independientes
- `SANITY_CHECK` El programa se detiene en caso de que no se cumpla alguna condición

En particular, consideramos relevante las versiones expuestas en 2.1.

[1] También expuesta en la clase teórica del lunes 26/8

2.1. Implementación

Como primer paso, se pueden generar dos versiones distintas del programa, variando la implementación de la siguiente función:

```
void*
paint(void *artist_ant, void *gridfn, colour_fn next_colour,
    rule_fn next_rotation, uint64_t iterations);
```

Se presume que la versión base demora mas tiempo en ejecutar que la alternativa -compilada con la opción `USE_TABLES`- puede completar la cantidad especificada de iteraciones en menos tiempo. Para comprobar si esta afirmación es cierta, utilizaremos las herramientas `time` y `gprof`.

2.2. Ejemplos

Listamos las opciones utilizando el comando `--help`

```
./tp0_if --help
  ./tp0_if -g <grid_spec> -p <colour_spec> -r <rule_spec> -t <n>
  -g --grid: wxh
  -p --palette: Combination of RGBYNW
  -r --rules: Combination of LR
  -t --times: Iterations. If negative, it's complement will be used.
  -o --outfile: output file. Defaults to stdout.
  -h --help: Print this message and exit
  -v --verbose: Version number

Compile with -DSANITY_CHECK to enable runtime checks
Compile with -DUSE_TABLES to execute ant operations in separate functions
Compile with -DUSE_COL_MAJOR to traverse the grid in column-major order
```

Medimos el tiempo en ejecutar diez mil operaciones en la menor grilla posible, y repetimos escalando la cantidad de operaciones

```
time -p ./tp0_if -g 1x1 -p RGBW -r LLLL -t ((10 * 1000))  > /dev/null
real 0.10
user 0.06
sys 0.02

time -p ./tp0_if -g 1x1 -p RGBW -r LLLL -t $((100 * 1000))  > /dev/null
real 0.18
user 0.15
sys 0.02

time -p ./tp0_if -g 1x1 -p RGBW -r LLLL -t $((1000 * 1000))  > /dev/null
real 1.41
user 1.20
sys 0.01
```

Repetimos, con una grilla significativamente mas grande

```
time ./tp0_if -g 1024x1024 -p RGBW -r LLLL -t $((10 * 1000))  > /dev/null

real 0m3.611s
user 0m3.072s
sys 0m0.032s

time ./tp0_if -g 1024x1024 -p RGBW -r LLLL -t $((100 * 1000))  > /dev/null

real 0m3.178s
user 0m2.784s
sys 0m0.012s

time ./tp0_if -g 1024x1024 -p RGBW -r LLLL -t $((1000 * 1000))  > /dev/null

real 0m3.414s
user 0m2.976s
sys 0m0.028s
```

3. Objetivos

Tal como se menciona arriba, se disponen de dos implementaciones de la versión `paint`. El objetivo del trabajo práctico estudiar cuál es el máximo *Speed Up* posible al optimizar dicha función, para después luego contrastar esta hipótesis con las mediciones realizadas sobre la nueva implementación.

Se espera que los siguientes puntos estén incluidos en el análisis del problema

- ¿Cómo varía el tiempo de ejecución a medida que se cambian los parámetros del programa?
- ¿Durante qué proporción de tiempo se puede aplicar la mejora?
- ¿Cuál es el máximo *Speed Up* posible?
- Análisis realizado con `gprof`
- Mediciones relevantes realizadas con `time`
- Comparaciones del tiempo de ejecución con distintos parámetros
- Cálculo del *Speed Up* global y local

4. Recursos

- Hormiga de Langton: `https://es.wikipedia.org/wiki/Hormiga_de_Langton`
- Formato PPM: `http://netpbm.sourceforge.net/doc/ppm.html`
- Imagemagick `https://imagemagick.org/index.php`
- GProf guía rápida `https://web.eecs.umich.edu/~sugih/pointers/gprof_quick.html`
- Manual gprof `https://linux.die.net/man/1/gprof`

5. Condiciones de entrega

5.1. Fechas de entrega

- 5/9/2019
- 19/9/2019
- Carátula especificando los datos y contacto de los integrantes del grupo (dirección de correo electrónico, *handle* de slack, ubicación del repositorio de código)
- Decisiones relevantes sobre la implementación y resolución
- Conclusiones con fundamentos reales
- Casos de prueba documentados
- Código fuente, si aplica
- Este enunciado