

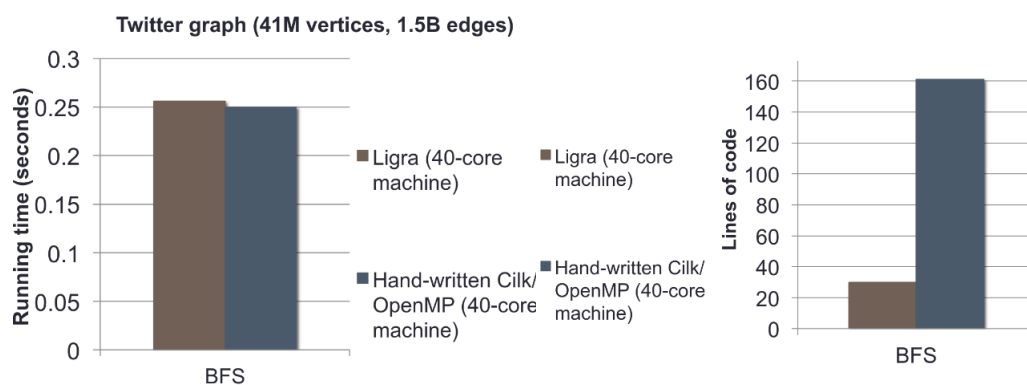
## 4.2. Práctica: Sobre el framework Ligra

### Ligra

Ligra (**L**ightweight **g**raph) es un framework desarrollado por Julian Shun, Guy E. Blelloch ([Shun., 2013](#)), entre otros desarrolladores del cómputo de alto rendimiento. El framework está diseñado para hacer uso de una arquitectura de memoria compartida para paralelizar algoritmos aplicados a gráficas. Está implementado en C++ y es poliformo; es decir, es posible usar distintas herramientas para paralelizar el código. Puede usarse OpenMP o Cilk .

### En 40 núcleos

La siguiente gráfica proporcionada por Julian y colaboradores muestra que en un equipo con 40 núcleos y procesando 41 millones de vértices con 1,5 billones de aristas tuvieron un tiempo de 0,25 segundos con Ligra y un poco menos con una implementación usando solo OpenMP , los tiempos se muestran en la Figura 4.2a. La ventaja radica en el número de líneas de código necesarias para la implementación de BFS , la Figura 4.2b hace la comparativa en este aspecto.



(a) Tiempo obtenido con 40 núcleos

(b) Gráfica que muestra la cantidad de líneas de código

## Actividad

En el apéndice B están las instrucciones para poder instalar el framework Ligra y se explica cómo generar archivos de texto que tienen la representación de una gráfica. La explicación de la notación se encuentra en la sección B.4.

Deberás realizar pruebas en tu computadora, antes de ello deberás obtener las especificaciones de hardware; es decir, el modelo de procesador, con cuantos núcleos cuenta el procesador así como todo lo que tú consideres lo suficientemente influyente al momento de realizar las ejecuciones que se te indican a continuación.

En un reporte en formato PDF deberás reportar a detalle lo antes mencionado, deberás también agregar una gráfica y una tabla como se muestra a continuación, reportando los tiempos que obtuviste en tu computadora.

A nivel de software también deberás mencionar aspectos que consideres que influyen al momento de realizar las pruebas.

Ejemplo de gráfica solicitada

A continuación se muestra en la Figura 4.3 una serie de pruebas realizadas en una computadora así como también el Cuadro 4.2 de datos correspondientes .



Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
5.01e-05	5.98e-05	5.1e-05	4.1e-05	5.6e-05
7.99e-05	0.000108	9.9e-05	8.01e-05	0.000102
4.89e-05	7.92e-05	6.39e-05	4.89e-05	6.51e-05

Cuadro 4.2: Tiempos obtenidos sin compartir recursos

Figura 4.3: Gráfica sin compartir recursos

Deberás realizar cinco pruebas de por lo menos tres ejecuciones cada una, el objetivo es obtener los tiempos de ejecución para el algoritmo BFS de **Ligra** , estas pruebas las realizarás con una gráfica que generes con las herramientas de **Ligra** , las especificaciones se encuentran en la memoria técnica, el número de vértices que debe tener esta gráfica es de un millón de vértices.

Además de las pruebas con un millón de vértices deberás individualmente probar crear gráficas de un mayor tamaño en tu computadora y reportar hasta que tamaño de gráfica fue capaz de crear tu computadora, deberás igual que con la gráfica de un millón de vértices reportar los tiempos obtenidos. Como parámetro considera que una computadora con cuatro núcleos fue capaz de crear hasta una gráfica de 11 millones sin que el sistema operativo dejará de funcionar o se trabara, considéralo en tus pruebas, pero recuerda que otros factores de hardware pudieron influir en esto. Procura ser cauteloso y recuerda que tener un respaldo de tus archivos es siempre una buena práctica, sobre todo al final del semestre.

Archivo a entregar

Para la entrega de esta práctica deberás enviar el archivo PDF reportando lo solicitado. Específicamente, el reporte debe cumplir con los siguientes requisitos:

- Detalles y especificaciones del hardware con el que cuenta la computadora en donde se realizaron las pruebas.
- Detalles y especificaciones del hardware en el que tu compañero realizó las mismas pruebas.
- Análisis de la diferencia de las pruebas realizadas entre las dos arquitecturas.
- Pruebas variando condiciones de software, esto debe ser alguna propuesta del alumno de cómo se vería afectado el rendimiento de **Ligra** .
- Gráficas y tablas de valores de todas las pruebas realizadas y descripción de las condiciones en las que se realizaron dichas pruebas, como se muestra en el Ejemplo 4.2.
- Referencias en formato APA si es el caso.

Punto extra

- Para obtener un punto extra sobre la calificación final de esta práctica deberás explicar con tus propias palabras en que consiste la representación gráfica que ocupa **Ligra** y dar un ejemplo con la secuencia de números y la construcción de la gráfica.

### 4.2.1. Actividad: Temperatura en los procesadores

A continuación se proporciona una guía para la instalación de `lm-sensors` para la lectura de la temperatura de los procesadores en el Sistema Operativo Ubuntu. Con esto se podrán hacer las lecturas para las pruebas solicitadas.

## Instalación

Para poder realizar el ejercicio se tiene que instalar `lm-sensors` a continuación se presentan los comandos para instalar y utilizar esta herramienta.

Como se ha venido manejando en todas las actividades los comandos son para utilizarlos en el sistema operativo Ubuntu.

El siguiente comando es para la instalación de la herramienta `sensors` que nos permite monitorear la temperatura de los procesadores con los que cuenta nuestro equipo.

```
umm@usuario:~$ sudo apt-get install lm-sensors
```

## Utilizar la herramienta

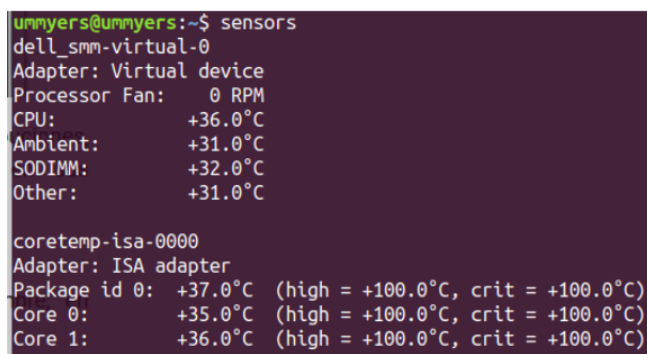
Con esta herramienta se pueden realizar diferentes pruebas. A continuación se en listan las que se proponen para la actividad.

- Registrar la temperatura de los procesadores cuando no se realiza ninguna otra tarea más que lo que el sistema operativo esté administrando.
- Registrar la temperatura de los procesadores ejecutando o abriendo alguna aplicación que sepa el usuario que requiere un esfuerzo a su equipo, por ejemplo, programas como **Anaconda**, **Android Studio**, entre otros.
- Realizar la misma prueba anterior pero con algún mecanismo externo, es decir, algún ventilador externo, los equipos de algunos alumnos requieren ya el uso de ventiladores externos, puede registrarse la temperatura haciendo uso de éste agente externo que influye en dicha lectura.
- Registrar la temperatura de los procesadores antes de ejecutar las pruebas propuestas en la práctica Sobre el **framework Ligra**, durante la ejecución y después, observando así el impacto de manejar y simular millones de vértices.
- Registrar la temperatura de los procesadores cuando se ejecuta el programa `busyCores.c` de la práctica Suma Paralela 1.3.

Para obtener la temperatura de los procesadores basta con ejecutar el siguiente comando en la terminal.

```
umm@usuario:~$ sensor
```

Se visualizará algo como lo que se muestra en la siguiente imagen.



```
ummyers@ummyers:~$ sensors
dell_smm-virtual-0
Adapter: Virtual device
Processor Fan:  0 RPM
CPU:            +36.0°C
Ambient:        +31.0°C
SODIMM:         +32.0°C
Other:          +31.0°C

coretemp-isa-0000
Adapter: ISA adapter
Package id 0:  +37.0°C (high = +100.0°C, crit = +100.0°C)
Core 0:        +35.0°C (high = +100.0°C, crit = +100.0°C)
Core 1:        +36.0°C (high = +100.0°C, crit = +100.0°C)
```

Figura 4.4: Resultados comando `sensors`

Con el fin de ilustrar brevemente los resultados que pueden obtenerse en este ejercicio se exponen algunos resultados que se obtuvieron.

La lectura registrada en la Figura 4.4 corresponde al equipo personal establecido para todas las prácticas y actividades del presente trabajo con el uso del ventilador externo y sin ningún programa inicializado en el sistema operativo. La siguiente captura de pantalla correspondiente a la Figura 4.5 muestra la temperatura sin el uso del ventilador externo.

```
unmyers@unmyers:~$ sensors
dell_smm-virtual-0
Adapter: Virtual device
Processor Fan: 0 RPM
CPU: +43.0°C
Ambient: +35.0°C
SODIMM: +34.0°C
Other: +33.0°C

coretemp-isa-0000
Adapter: ISA adapter
Package id 0: +46.0°C (high = +100.0°C, crit = +100.0°C)
Core 0: +45.0°C (high = +100.0°C, crit = +100.0°C)
Core 1: +46.0°C (high = +100.0°C, crit = +100.0°C)
```

Figura 4.5: Resultados del comando `sensors` sin ventilador externo

Dada la propuesta de ejecutar un programa que requiera un poder considerable del equipo, las siguientes lecturas de la Figura 4.6 corresponden al momento de estar utilizando Anaconda en el equipo Dell.

```
unmyers@unmyers:~$ sensors
dell_smm-virtual-0
Adapter: Virtual device
Processor Fan: 0 RPM
CPU: +36.0°C
Ambient: +31.0°C
SODIMM: +32.0°C
Other: +31.0°C

coretemp-isa-0000
Adapter: ISA adapter
Package id 0: +37.0°C (high = +100.0°C, crit = +100.0°C)
Core 0: +35.0°C (high = +100.0°C, crit = +100.0°C)
Core 1: +36.0°C (high = +100.0°C, crit = +100.0°C)
```

Figura 4.6: Resultados comando `sensors` ejecutando Anaconda

Dicha actividad puede realizarse en cualquier altura el semestre, sin embargo, resulta más interesante ver como la práctica Sobre el framework Ligra 4.2 puede repercutir en la temperatura de los procesadores, que solo ver el impacto de algunas aplicaciones, IDE's o programas en ellos.

## Apéndice B

# Memoria técnica del framework Ligra

### B.1. Pre-requisitos

Para poder hacer uso de Ligra se debe contar con la siguiente lista de requerimientos:

- Compilador C++
- API OpenMP
- Repositorio de Ligra clonado
- Variable de entorno configurada de OpenMP

A continuación, se presentan las instrucciones para hacer la instalación de las herramientas necesarias, para el sistema operativo Ubuntu en su versión 18.04.4 LTS, los comandos sirven igualmente para Debian 3.16.43.

#### 1. Compilador y OpenMP

A continuación, se describen los pasos para instalar el compilador de C y para poder usar OpenMP ; sin embargo, se esperaría que el equipo ya cuente con los dos primeros puntos.

Para instalar el compilador y poder trabajar con OpenMP basta con los siguientes comandos:

```
umm@usuario:~$ sudo apt-get install g++
umm@usuario:~$ sudo apt-get install gcc-multilib
```

#### 2. Repositorio

Para clonar el repositorio es necesario contar con la instalación de Git previamente. Se da por hecho que ya se tiene instalado Git.

El comando para clonar el repositorio es el siguiente, recuerda colocarte dentro de la carpeta donde quieres que se clone:

```
umm@usuario:~$ git clone https://github.com/jshun/ligra.git
```

La URL del repositorio es la siguiente: <https://github.com/jshun/ligra.git>

#### 3. Variable de entorno y compilación

Para la configuración de la variable de entorno, basta con el siguiente comando

```
umm@usuario:~/Documentos/.../ligra$ export OPENMP=1
```

## B.2. Compilación

Para usar el framework se hará uso de **Make**, la herramienta de gestión de dependencias.

Para poder hacer uso de **Make**, debes tener instalado previamente esta herramienta, para hacerlo basta con el siguiente comando:

```
umm@usuario:~/Documentos/...$ sudo apt-get install make
```

```
umm@usuario:~/Documentos/.../ligra$ make -j
```

Después de haber establecido que se trabajará con OpenMP y la compilación mediante **Make** se espera un resultado como lo mostrado en la Figura B.1.

```
g++ -std=c++14 -fopenmp -march=native -O3 -DOPENMP -DBYTERLE -o BC BC.C
g++ -std=c++14 -fopenmp -march=native -O3 -DOPENMP -DBYTERLE -o BellmanFord
BellmanFord.C
g++ -std=c++14 -fopenmp -march=native -O3 -DOPENMP -DBYTERLE -o Components C
omponents.C
g++ -std=c++14 -fopenmp -march=native -O3 -DOPENMP -DBYTERLE -o Components-S
hortcut Components-Shortcut.C
g++ -std=c++14 -fopenmp -march=native -O3 -DOPENMP -DBYTERLE -o Radii Radii.
C
g++ -std=c++14 -fopenmp -march=native -O3 -DOPENMP -DBYTERLE -o PageRank Pag
eRank.C
g++ -std=c++14 -fopenmp -march=native -O3 -DOPENMP -DBYTERLE -o PageRankDelt
a PageRankDelta.C
g++ -std=c++14 -fopenmp -march=native -O3 -DOPENMP -DBYTERLE -o BFSCC BFSCC.
C
```

Figura B.1: Compilación exitosa

### Para la ejecución

Los archivos de ejemplos se encuentran en la carpeta **inputs**, para ejecutar los ejemplos de BFS y Bellman Ford con las gráficas ya construidas por los desarrolladores de Ligra se usan los siguientes comandos:

```
umm@usuario:~/Documentos/.../ligra/apps$ ./BFS -s ../inputs/
rMatGraph_J_5_100
umm@usuario:~/Documentos/.../ligra/apps$ ./BellmanFord -s ../inputs/
rMatGraph_J_5_100
```

Con la bandera **-rounds** se le indica a Ligra cuantas veces deberá ejecutar el algoritmo indicado, se reportará el tiempo de cada ejecución seguido de la leyenda **Running time**. Por default se ejecuta tres veces el algoritmo, es decir que si no se indica nada con la bandera **-rounds** un número en específico, se ejecutará tres veces el algoritmo.

Ejemplo del uso de bandera

```
umm@usuario:~/Documentos/.../ligra/apps$ ./BFS -rounds 2 -s ../inputs/
rMatGraph_J_5_10
```

Si deseas borrar todos los ejecutables de Ligra debes usar el siguiente comando:

```
umm@usuario:~/Documentos/.../ligra/apps$ make clean
```

## B.3. Generador de gráficas

Ligra cuenta con un generador automático de gráficas, esto significa que podemos hacer que Ligra genere el archivo de texto para la representación para poder hacer pruebas. A continuación, se muestran los pasos a

realizar para poder hacer uso de esta utilidad de Ligra .

Primero que nada debemos compilar los archivos que se encuentran en la carpeta `utils`.

```
umm@usuario:~/Documentos/.../ligra/utils$ make
```

Con ello podremos hacer uso de los ejecutables para generar una gráfica de la cardinalidad de vértices y aristas que queramos. Sin embargo Ligra cuenta con varios formatos para representar gráficas, se presentan los de interés.

### Banderas para el generador

Cuando creamos una gráfica de tipo `randLocalGraph` podemos indicar el número de aristas que tiene la gráfica, con la bandera `-m`, después de ello recibe el número de vértices y el nombre del archivo de salida.

**Ejemplo** Se crea una gráfica con una arista y dos vértices

```
umm@usuario:~/Documentos/.../ligra/utils$ ./randLocalGraph -m 1 2 Gra_5
```

Para generar una gráfica con los vértices que quieras se puede usar el siguiente comando:

```
umm@usuario:~/Documentos/.../ligra/utils$ ./rMatGraph <n> rMat_100
```

Dónde  $n$  es el parámetro que especifica el número de vértices y `rMat_100` es el nombre del archivo de salida que se creará. El comando anterior nos da una gráfica de tipo `AdjacencyGraph`, para más información consulta ([Shun, s.f.](#)). Sin embargo, es suficiente para crear las gráficas con el tamaño solicitado.

## B.4. Representación de una gráfica en Ligra

Para manipular algoritmos sobre Teoría de Gráficas, Ligra debe tener alguna forma de representación de este modelo matemático. Los autores retomaron la representación de las gráficas desde el siguiente link <http://www.cs.cmu.edu/~pbbs/benchmarks/graphIO.html>

El formato de representación es el siguiente:

```
AdjacencyGraph
<n >
<m >
<o1 >
...
<o(n-1)>
<e0 >
<e1 >
...
<e(m-1) >
```

Donde  $n$  es el número de vértices y  $m$  el número de aristas.