Taller de Sistemas Operativos, 2020
Taller 03

Profesor: Gabriel Astudillo Muñoz
Escuela de Ingeniería Civil Informática
Universidad de Valparaíso

1 Objetivo

Implementar un programa que llene un arreglo de números enteros y luego los sume. Ambas tareas se deben realizar en forma paralela, implementadas con OpenMP.

2 Instrucciones

Su trabajo deberá ser entregado en el servidor en su cuenta en **`github.com`**. El nombre del repositorio debe ser **`TSSOO-taller03`** y deberá tener la siguiente estructura:

```
+ /TSSOO-taller03
     - README.md
     - informeT03-Apellido1Apellido2Nombre.pdf
     - Makefile
     + src/
          + include/
               - global.hh
               - checkArgs.hpp
          - Makefile
          - main.cc
```

El archivo **`README.md`** es un archivo en markdown, donde se debe explicar el diseño de su solución, funciones utilizadas, etc. Además, debe especificar el autor y su correo institucional. El código de su proyecto debe estar ordenado, utilizar estructuras de datos apropiadas e implementado en C++ 2014 o superior, y debe responder a un diseño previo, el que debe ser realizado y explicado en un informe técnico, que debe estar en su repositorio y cuyo nombre debe ser **`informeT03-Apellido1Apellido2Nombre.pdf`**. Este reporte deberá contener el diseño, pruebas, validación y análisis de los resultados entregados y la metodología utilizada para enfrentar el desafío.

3 Trabajo a realizar

Crear un programa que esté compuesto de dos módulos. Uno que llene un arreglo de números enteros aleatorios del tipo **`uint32_t`** en forma paralela y otro que sume el contenido del arreglo en forma paralela. Debe hacer pruebas de desempeño que generen datos que permitan visualizar el comportamiento del tiempo de ejecución de ambos módulos dependiendo del tamaño del problema y de la cantidad de threads utilizados.

Para generar números aleatorios, debe utilizar funciones que sean *thread safe* para que observe una mejora en el desempeño de su programa.

3.1 Forma de uso:

```
./sumArray -N <nro> -t <nro> -l <nro> -L <nro> [-h]
```

Parámetros:

```
     -N   : tamaño del arreglo.
     -t   : número de threads.
     -l   : limite inferior rango aleatorio.
     -L   : límite superior rango aleatorio.
     [-h] : muestra la ayuda de uso y termina.
```

3.2 Ejemplo de uso:

1) Crea un arreglo de 1000000 posiciones, con 4 threads. Los números enteros aleatorios están en el rango [10,50]

```
./sumArray -N 1000000 -t 4 -l 10 -L 50
```

2) Muestra la ayuda y termina

```
./sumArray -h
./sumArray
```