IMT2112 Semestre 2022-2
Tarea 3

Elwin van 't Wout

October 7, 2022

Introducción

El algoritmo PageRank era diseñado para calcular la importancia de páginas de web en el internet y es comunmente usado por buscadores como Google. El web es representado como un grafo, con cada página un nodo y los hyperlinks las conexiones (aristas). A su vez, el grafo es representado por una matriz y, después de algunas manipulaciones, el vector propio principal entrega la importancia de las páginas. Una parte esencial y costosa en este algoritmo es encontrar el valor propio más grande de una matriz. El 'método de potencia' (*power iteration*) es un algoritmo iterativo que aproxima el vector propio más grande de una matriz.

El método de potencia funciona como siguiente. Dado una matriz $A \in \mathbb{R}^{n\times n}$ diagonalizable, la iteración

$$\mathbf{b}_{k+1} = \frac{A\mathbf{b}_k}{||A\mathbf{b}_k||}$$

converge al vector propio de A que corresponde al valor propio más grande. La secuencia

$$\mu_k = \frac{\mathbf{b}_k^T A\mathbf{b}_k}{\mathbf{b}_k^T\mathbf{b}_k}$$

converge al valor propio más grande. El método debe ser inicializado con un vector $\mathbf{b}_0$.

Tarea

Esta tarea contempla una implementación del método de potencia con MPI.

1. Generen una matriz $A \in \mathbb{R}^{n\times n}$ en Python, usando la función siguiente:

```
def generate_matrix(dim):
    from scipy.stats import ortho_group
    from scipy.sparse import spdiags
    a = ortho_group.rvs(dim, random_state=0)
```

```
b = np.linspace(1., 10., dim)
return a @ spdiags(b, 0, dim, dim) @ a.T
```

y guarden la matriz en un archivo. Además usen las primeras filas de este archivo para guardar las dimensiones de la matriz.
Observación: esta función genera una matriz con valores propios reales entre 1 y 10.

2. Todas las instrucciones siguientes deben ser implementadas en C++.

3. Leen el tamaño de la matriz desde el archivo y calculen el tamaño de cada bloque de la matriz en cada proceso.

   - La matriz debe ser particionado por bloques de filas.
   - El código debe funcionar para cualquier número de procesos y dimensión de la matriz. En específico, también cuando la dimensión no es un múltiplo del número de procesos.

4. Inicialicen el vector $\mathbf{b}_0$ con valores igual a uno. Este vector también debe ser particionado sobre los procesos.

5. Leen la parte de la matriz que corresponde desde el archivo. Cada proceso sólo guarda su parte de la matriz. Guarden los bloques en un arreglo 1D, para lo cual deben usar *dynamic memory allocation*.

   - Hay un ejemplo de leer archivos en C++ disponible en Canvas.
   - Más información de crear arreglos dinámicos: `https://cplusplus.com/doc/tutorial/dynamic/`.

6. Implementen el método de potencia con MPI.

   - Se puede usar un número fijo de iteraciones.
   - Nunca se puede enviar elementos de la matriz entre procesos.
   - Todas las instrucciones matemáticas, tales como multiplicación matriz por vector, productos internos y normas, deben ser paralelizados sobre todos los procesos.

7. Calculen el error del método, es decir, la diferencia entre el valor propio máximo estimado y el valor exacto, lo cual es 10 para esta matriz específica.

8. Corren el algoritmo con distintos números de procesos y miden el tiempo de cómputo. Discuten la diferencia en tiempo de cálculo cambiando el número de procesos.

   - Deben correr el código en el clúster de Ingeniería UC, usando la cola de trabajo `full`. ¡No pueden usar el nodo de login para hacer cálculos!

- Imprimen el nombre del procesador en cada proceso de MPI.
- Usen una dimensión de la matriz y un número de iteraciones adecuada, tal que el código demora un par de minutos.
- Calculen el speedup y eficiencia paralela y analicen ambos *strong scaling* y *weak scaling*[1].
- ¿Qué pasa si pides tantos procesos que SLURM debe asignar dos nodos distintos?

Evaluación

Entreguen todo el código de C++, el script de SLURM, y el informe (en pdf) en una mapa comprimida a través de Canvas.

Los reglamientos del curso se puede encontrar en Canvas. Se destaca que las tareas deben ser hechas de forma individual y deben cumplir las políticas de uso del clúster de Ingeniería UC.

Sugerencias

Si el código quedó pegado, por ejemplo en un deadlock, pueden terminar la tarea con `scancel 12345` con el número cambiado por el identificador del job. Pueden revisar la cola de trabajo con `squeue`.

En BASH, y por lo tanto también en el script de SLURM, el comando `date` imprime la fecha y hora y el comando `time mpirun -np 2 a.out` imprime el tiempo de cálculo de MPI.

Ojo que en el clúster no pueden elegir el nodo de cómputo: SLURM lo hace para ti. Cada nodo tiene su propia características y uno es más rápido que el otro.

Algunos comandos útiles de BASH para usar en el terminal/console:

- Ordenar los archivos por fecha: `ls -ltr`
- Ver el tamaño de archivos: `du -shc *`
- Mostrar las últimas líneas de un archivo: `tail -f salida.txt` (salir con CTRL+c)

En el clúster de Ingeniería el comando `MPI_Initialize()` da un error tipo `sys.c:618 UCX ERROR shmget(size=2097152 flags=0xb80) for ucp_am_bufs failed: Operation not permitted, please check shared memory limits by 'ipcs -l'.`

[1] Se puede copiar los tiempos de cómputo a Excel o Python para hacer este análisis.

Este es un problema conocido de esta versión de MPI y es más bien un warning: el código sigue igual.

El comando `#SBATCH --output=output_%j.txt` guarda el número del job en el nombre de archivo.

Ver `https://cluster.ing.puc.cl` para más información sobre el clúster de Ingeniería UC.