Applied High Performance Computing

Proyecto

Posgrado

2025-I

Profesor José Fiestas

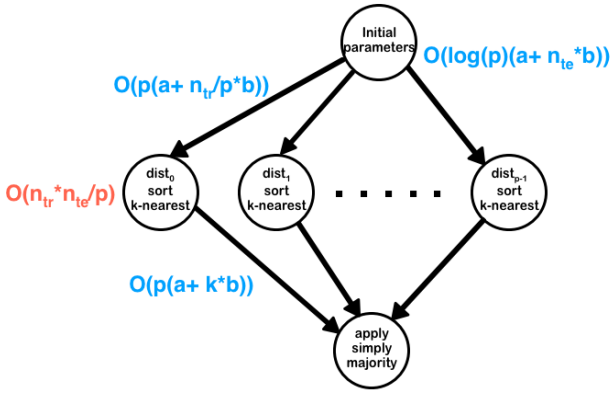
Indicaciones generales:

- se presenta un conjunto de proyectos para que el estudiante escoja aquel que va a desarrollar
- El proyecto se desarrollará en grupos de 2 o 3 estudiantes. Los **nombres** de los integrantes deberán aparecer al inicio del **informe de proyecto**
- Los archivos de proyecto se subirán directamente a Canvas (**Proyecto HPC**)
- La entrega parcial del proyecto será hasta el **sábado 18 de octubre**. La entrega final hasta el **sábado 29 de noviembre**
- La presentación final y discusión sobre el proyecto será el **sábado 06 de diciembre o jueves 11 de diciembre** en clase
- La sección 8 describe **Indicaciones importantes para el proyecto**, y la sección 9, la **Rúbrica**

---

1 KNN

KNN es un algoritmo de clasificación supervisada (aunque también puede aplicarse a regresión) que clasifica un punto nuevo según las etiquetas de sus k vecinos más cercanos en el espacio de características. En paralelo, cada elemento de testeo puede ser calculado independientemente del resto con la fracción de la data de prueba que tiene cada proceso.



Paralelice el código adjunto *knn_digits_sec.py* [1], que utiliza la conocida librería *load_digits* para clasificar dígitos.
El desarrollo y **análisis de resultados** debe contener lo siguiente:

a) Elabore el código en Python incluyendo las directivas de comunicación (comm.bcast, comm.scatter, comm.gather)[2] de acuerdo al criterio mostrado en el DAG

b) Registre el desarrollo del código en por lo menos 3 pasos (versiones beta). En caso utilice una fuente externa, mostrar la validación del código

c) Mida los tiempos de **ejecución**, **cómputo** y **comunicación** (represéntelas gráficamente), así como la **precisión** del modelo (accuracy). Incremente tanto procesos (p) como datos (n). Utilice la complejidad teórica para **validar** el resultado obtenido. Describa como normaliza la expresión teórica para que pueda aproximar los experimentos. ¿Cuál estima, sería la cantidad **óptima de procesos**?

d) Derive la gráfica de **speedup** de los datos anteriores y genere una gráfica de **FLOPs vs. p** (FLOP por segundo). Obtenga los FLOP de la región paralelizable (entrenamiento), basada en la fórmula de la **distancia euclidiana entre dos puntos**.

e) Analice la escalabilidad del algoritmo con un tamaño variable del problema

Bibliografía:
[1] https://drive.google.com/file/d/1wTBSh7bwT1QVvce0pFMv5tcCWsg4-HeK/view?usp=sharing
[2] https://mpi4py.readthedocs.io/en/stable/

2 TSP (Travelling Salesman Problem)

El Problema del Viajante (TSP) plantea lo siguiente: ¿Cuál es la ruta más corta posible que permite a un viajante visitar un conjunto de ciudades una sola vez y regresar a la ciudad de origen?.

Es un problema clásico de optimización NP-difícil, lo que significa que no hay una solución eficiente conocida para todos los casos. Muchos problemas en ciencia de datos requieren encontrar la mejor solución entre muchas posibles (por ejemplo: rutas, horarios, combinaciones). Ello deriva en aplicaciones directas como:

- Logística y entrega de productos (Amazon, Rappi, FedEx)
- Rutas de drones y robots autónomos
- Secuenciación de genomas (ordenar fragmentos de ADN)
- Diseño de circuitos electrónicos, conexión satelital, en fibra óptica

En grandes volúmenes de datos, el orden en que se procesan tareas puede modelarse como un problema similar al TSP. Esto mejora la eficiencia de: Pipelines de datos, Procesamiento distribuido

El código adjunto[(1)] resuelve el problema del agente viajero en paralelo utilizando el método *branch and bound* discutido en clase
El código aplica el algoritmo a matrices de distancia entre ciudades (ver data adjunta)
El proyecto debe contener lo siguiente:
a) Elabore un PRAM del algoritmo y determine la complejidad teórica del mismo
b) Ejecute mediciones de tiempo con el código y compárelas con la teórica, para distinto número de procesos y tamaño del problema
c) Desarrolle un software de análisis de performance que permita obtener métricas de performance. El código debe generar la representación gráfica de las métricas y llegar a conclusiones sobre la escalabilidad del mismo.

Bibliografía
[1] TSP code

3 Expresiones regulares

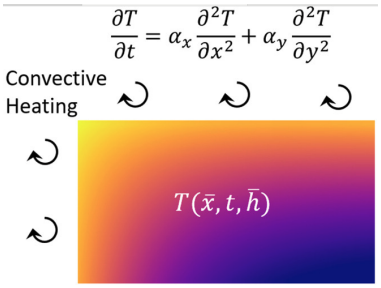
Expresiones regulares (regex) son una forma de representar un conjunto de cadenas de caracteres que satisfacen ciertas reglas de construcción (gramáticas). E.g. dado el alfabeto $\Sigma = \{0, 1\}$, la expresión regular $01(01)^*$ acepta cadenas como {01,0101,010101,...}. Expresiones regulares se usan frecuentemente en algoritmos de manipulación de strings, de análisis lexicográfico y sintáctico, entre otros. Un ejemplo de ello es NetKat, un lenguaje de programación de redes [1]
En Ciencia de Datos, regex es un poderoso utilitario que sirve para manipular patrones en texto, en operaciones de búsqueda, matcheo, reemplazo, splitting, extraccion de datos, entre otros.

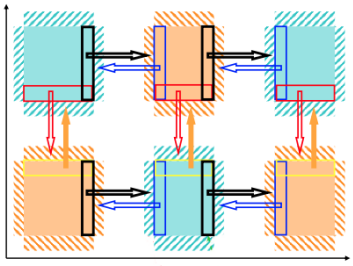
Un software de reconocimiento de expresiones regulares es de mucha utilidad, especialmente si es escalable. PAREM [2] es un ejemplo de un sotware de expresiones regulares rápido y escalable, con un speedup lineal con respecto al número de procesos. Este algoritmo particiona la cadena (input), para luego combinar los resultados parciales obtenidos (ver paper).
Se adjunta en el proyecto, una version de PAREM [(1)], desarrollada con OpenMP (visto en clase)
El proyecto debe contener lo siguiente:
a) Elabore un PRAM del algoritmo y determine la complejidad teórica del mismo
b) Ejecute mediciones de tiempo con el código y compárelas con la teórica, para distinto número de procesos y tamaño del problema
c) Desarrolle un software de análisis de performance que permita obtener métricas de performance. El código debe generar la representación gráfica de las métricas y llegar a conclusiones sobre la escalabilidad del mismo.

Bibliografía:
[1] PAREM code
[2] arxiv.org/pdf/1412.1741.pdf

4 Ecuación de calor

Es un caso de solución de un sistema de ecuaciones diferenciales parciales para resolver la ecuacion de transferencia de calor. Se utiliza Machine Learning para hacer proyecciones en transferencia de calor, útil para manufactura y aplicaciones en ingeniería.
En este proyecto debe realizar un análisis de performance con el código MPI (adjunto)[(1)] que resuelve la ecuación de calor (EDP, vista en clase) en paralelo.





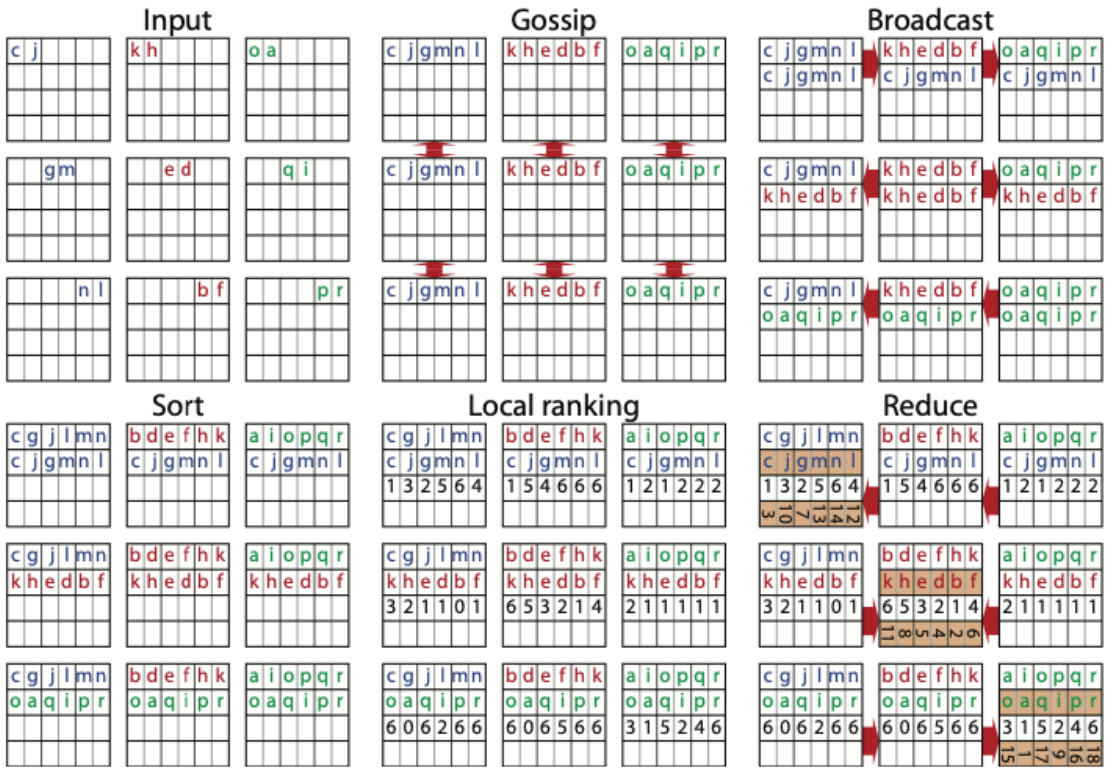
El proyecto debe contener lo siguiente:
a) Elabore un PRAM del algoritmo y determine la complejidad teórica del mismo
b) Ejecute mediciones de tiempo con el código y compárelas con la teórica, para distinto número de procesos y tamaño del problema
c) Desarrolle un software de análisis de performance que permita obtener métricas de performance. El código debe generar la representación gráfica de las métricas y llegar a conclusiones sobre la escalabilidad del mismo.

Bibliografía

[1] heat

5 Ordenamiento eficiente por ranking

Se pide desarrollar un código en paralelo para el algoritmo de ordenamiento por ranking visto en clase



El proyecto debe contener lo siguiente:
a) Elabore un PRAM del algoritmo y determine la complejidad teórica del mismo
b) Ejecute mediciones de tiempo con el código y compárelas con la teórica, para distinto número de procesos y tamaño del problema
c) Desarrolle un software de análisis de performance que permita obtener métricas de performance. El código debe generar la representación gráfica de las métricas y llegar a conclusiones sobre la escalabilidad del mismo.

Bibliografía:
[1] RankingSort code

6 Evolución galáctica

Simulaciones de evolución galáctica permiten predecir teóricamente el comportamiento dinámico de estrellas en galaxias y cúmulos globulares. El modelo requiere conjuntos de objetos (N

estrellas) de una cantidad muy grande, billones de estrellas en la realidad, que exigen una simulación lo más realistica posible, pero cuyos resultados sean accesibles en un tiempo mesurado.

Esta propuesta se basa en el trabajo hecho por el grupo Astrofísica de la Universidad de Heidelberg, Alemania. Para ello, se cuenta con el código hibrido de NCuerpos PhiGPU [1].

El proyecto debe contener lo siguiente:
a) Elaborar un PRAM del algoritmo basandose en el algoritmo de NCuerpos discutido en clase y determinar la complejidad teórica del mismo
b) Ejecutar mediciones de tiempo con el codigo y compararlas con la teórica, para distinto número de procesos y tamaño del problema. El código cuenta con un Makefile que se ejecuta con: *make cpu-4th* Para modificar el tamaño del problema (cantidad de elementos) utilice en código *gen-plum.c* en la misma carpeta, modificando la variable N
c) Desarrolle un software de análisis de performance que permita obtener métricas de performance. El código debe generar la representación gráfica de las métricas y llegar a conclusiones sobre la escalabilidad del mismo.

Bibliografía
[1] phiGPU code
[2] Paper2011

7 Indicaciones importantes para la entrega del proyecto

7.1 Entrega Parcial (P1)

La primera entrega consiste en desarrollar el punto (a) del proyecto:
a) Elabore un PRAM del algoritmo y determine la complejidad teórica del mismo
Se calificará según la rúbrica (abajo)

7.2 Informe Final de proyecto (P2)

En la segunda entrega, se pide realizar un informe de los resultados del proyecto, que contenga lo siguiente:

- **Nombre del proyecto**, y de los **integrantes**
- **Introducción:** Describa la importancia de
- **Método:** Describa el algoritmo usado y la optimización. Muestre el PRAM si se solicita.
- **Resultados:** Describa los resultados obtenidos y plantee mejoras a la solución.

7.3 Entrega grupal

El proyecto se realizará en forma **grupal (2 o 3 integrantes)**
El proyecto se subirá a Canvas (Proyecto HPC)
Presente el documento de proyecto en formato **.pdf**. Puede convertir un documento a pdf o usar una plantilla de Plantillas Latex . No olvide adjuntar el **código de programación** o **link a Github**.

8 Rúbrica

Se utilizará la siguiente rúbrica para la calificación del proyecto

Criterio	Logrado	Parcialmente Logrado	No Logrado
PRAM y análisis de tiempos	PRAM y análisis realizado son correctos (`6 pts`)	Existen algunos errores menores en el PRAM y/o análisis de tiempos (`3 pts`)	Existen muchos errores en la solución mostrada(`0pts`).
Software performance	El software es correcto y responde a los requerimientos del proyecto (`4 pts`)	El software contiene errores menores (`2 pts`).	El software no responde a los requerimientos del proyecto (`0pts`).
Presentación escrita (informe) y trabajo en grupo	Describe los puntos del informe en forma ordenada y satisfactoria en un grupo de 3 integrantes (`6 pts`)	No describe en forma adecuada todos los puntos del informe(`3 pts`).	No describe en forma adecuada ningún punto del informe (`0pts`).
Presentación oral	Se presentó el proyecto en forma ordenada y clara (`4 pts`)	No se demostró un dominio total del contenido del proyecto durante la presentación (`2 pts`).	No se presentó conocimiento del proyecto durante la presentación (`0pts`).