

Indicaciones generales:

- se presentan dos tipos de proyecto: **(A)** Desarrollo de código en paralelo y **(B)** Desarrollo de código de análisis de Performance
  - 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 [www.gradescope.com](http://www.gradescope.com) (**Proyecto Laboratorio**)
  - La entrega de proyecto será hasta el **viernes 12 de julio**, a medianoche
  - La presentación oral será el **lunes 15 de julio** y **miércoles 17 de julio**
  - La sección 8 describe **Indicaciones importantes para el proyecto**, y la sección 9, la **Rúbrica**
-

## 1 Simulación de plasma (Proyecto tipo B)

En este proyecto se trata de analizar el performance de un código Particle in Cell (PIC) que es usado para simulaciones del plasma en la ionosfera y sus posibles efectos en el clima. Los modelos teóricos se basan en la solución de un sistema de ecuaciones que describen la función de distribución del sistema, conocida como la ecuación de Vlasov (vea ecuaciones discretizadas en Unidad 6.1)

$$\left( \partial_t + \frac{\mathbf{P}}{m_s \gamma} \cdot \nabla + \mathbf{F}_L \cdot \nabla_{\mathbf{P}} \right) f_s = 0,$$

Esta ecuación describe la dinámica del plasma, considerando efectos eléctricos y magnéticos, de acuerdo a sus densidades. Como resultado, modelos físicos como el mencionado, logran elaborar mapas de la ionosfera como el siguiente Permitiendo realizar pronósticos de efectos

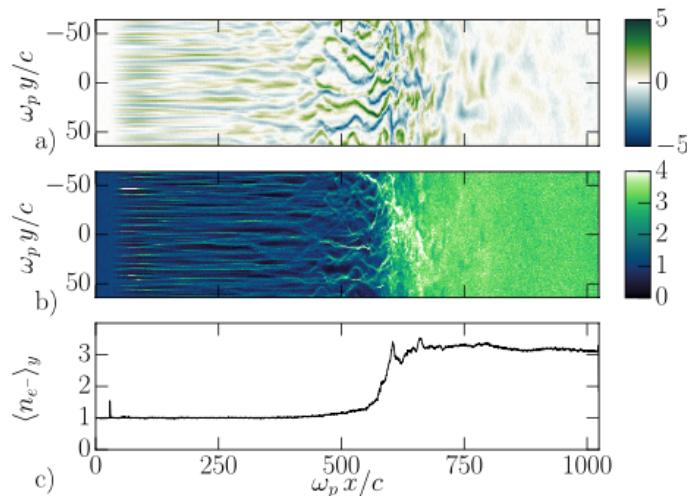


Figure 20: Snapshot at  $t = 1000 \omega_p^{-1}$ . a) Weibel generated magnetic field  $B_z$  in units of  $B_0 = m_e \omega_p / e$ .  
b) Electron density in units of  $n_0$ . c) Electron density (in units of  $n_0$ ) averaged along the  $y$ -direction.

en la evolución del clima de una región dependiendo de su latitud y condiciones propias del lugar.

El proyecto no requiere demostrar la física detrás de estos fenómenos. Se trata de analizar el **performance** y **escalabilidad** del código **PIC Smilei**, adjunto, cuyo diseño se ilustra en el siguiente diagrama UML

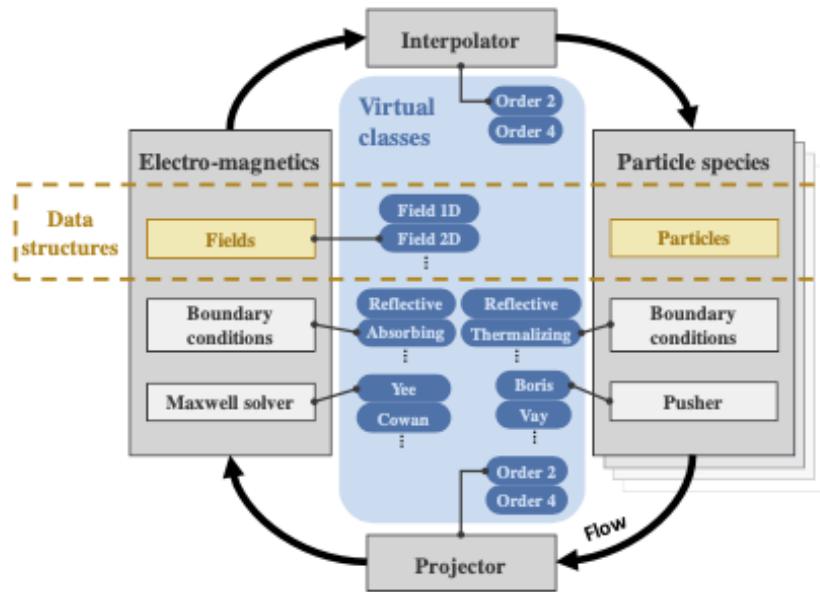


Figure 2: C++ flow, classes and data structure in SMILEI.

Desarrolle el siguiente análisis de performance de Smilei en el cluster Khipu:

- Reproduzca el setup del tutorial Smilei, así como el caso: Thermal Plasma
- Desarrolle el caso Performance/Parallel Computing. Muestre los resultados obtenidos al seguir el tutorial
- Determinar la complejidad teórica del código (utilice el paper y documentacion) y compararla con mediciones de tiempo, para distinto número de procesos y tamaño del problema
- Medir la velocidad del algoritmo, y representarla en gráficas. Analizar la escalabilidad del software
- Optimización:** Desarrolle el caso Performance/Patch Arrangement y muestre los resultados obtenidos.

## Bibliografía

- [1] <https://smileipic.github.io/Smilei/index.html>
- [2] <https://www.sciencedirect.com/science/article/pii/S0010465517303314?via%3Dihub>

## 2 TSP (Proyecto tipo A)

Resuelva el problema del agente viajero en paralelo utilizando el método *branch and bound* discutido en clase

Se trata de un vendedor que debe minimizar el recorrido entre n ciudades. Puede empezar en cualquiera de ellas y terminar en la misma luego del recorrido. Cada ciudad debe ser visitada solo una vez.

TSP se aplica a diversos problemas como DNA y genomas, conexiones de electricidad entre ciudades, conexión satelital, conexiones en fibra óptica, VLSI (Very Large System Integration)

Escoja una de las siguientes aplicaciones del algoritmo en paralelo:

- Aplique el algoritmo TSP desarrollado a un set VLSI del siguiente enlace: VLSI datasets. Escoja el set adecuado para que los resultados obtenidos sean claros. En caso los recursos disponibles (hardware) no sean suficientes para completar algunos de los casos VLSI, puede reducir la data original
- Utilice 10 puntos geográficos en sendos distritos de Lima y calcule las distancias entre ellos. Encuentre la ruta más corta al recorrer los 10 puntos. Compare los resultados escogiendo ahora 20 puntos en Lima (puede ubicar más de un punto en un distrito)

Puede escribir el código desde cero o utilizar algún código fuente para solucionar el problema, pero necesita validarla, mostrando su **precisión** antes de usarlo para el caso planteado.

El proyecto debe contener lo siguiente:

- a) Escoja el paradigma adecuado, elabore un PRAM y desarrolle un código en paralelo en C++
- 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) Realice análisis de tiempos y compare los resultados con la complejidad teórica esperada
- d) **Optimización:** Analice la escalabilidad del algoritmo con un tamaño variable del problema

### 3 Evolución galáctica (Proyecto tipo B)

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 realista 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 híbrido de NCuerpos PhiGPU [1].

El proyecto debe contener lo siguiente:

- a) Elaborar un PRAM del algoritmo basándose en el algoritmo de NCuerpos discutido en clase y determinar la complejidad teórica del mismo
- b) Ejecutar mediciones de tiempo con el código 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) Medir la velocidad del algoritmo, y representarla en gráficas. Analizar la escalabilidad del software
- d) **Optimización:** Desarrollar un software de análisis de performance que se adapte al código utilizado y permita obtener métricas de performance. El código debe llegar a conclusiones sobre la escalabilidad del mismo.

#### Bibliografía

- [1] phiGPU code
- [2] Paper2011

## 4 Expresiones regulares (Proyecto tipo A)

Expresiones regulares 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  $\{01, 0101, 010101, \dots\}$ . 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]

Un software de reconocimiento de expresiones regulares es de mucha utilidad, especialmente si es escalable. PAREM [2] es un ejemplo de un software de expresiones regulares rápido y escalable, con un speedup lineal con respecto al número de procesos. Este algoritmo partitiona la cadena (input), para luego combinar los resultados parciales obtenidos (ver paper).

El proyecto debe contener lo siguiente:

- a) Desarrollar un código en C++, correspondiente al paradigma apropiado de paralelismo
- b) Registro del desarrollo del código en por lo menos 3 pasos (versiones beta)
- c) Realice el análisis de tiempos, comparando mediciones con la complejidad teórica del código. Para distinto número de procesos y tamaño del problema
- d) **Optimización:** Desarrolle un software que realice el análisis de tiempos en forma automática

### Bibliografía:

- [1] arxiv.org/pdf/1412.1741.pdf

## 5 Cluster RPI (Proyecto tipo B)

Se pide instalar y configurar un cluster de ordenadores Raspberry PI, y hacer pruebas de escalabilidad en el mismo



El proyecto debe contener lo siguiente:

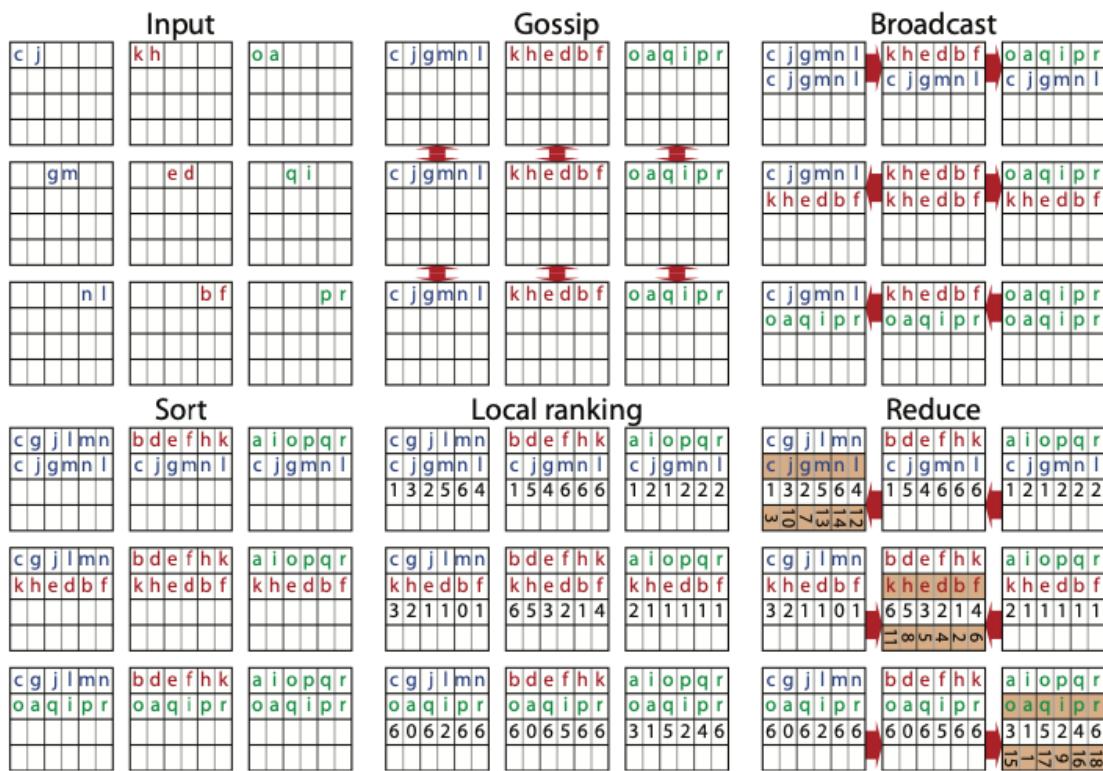
- a) Instale un cluster de Raspberry PIs y complete su configuración para la ejecución de tareas en forma distribuida (paralelismo). Utilice el software phiGPU code (ver proyecto 4 para su funcionamiento).
- b) Realice un informe describiendo las partes y el funcionamiento del cluster
- c) Efectúe pruebas de rendimiento en el cluster, utilizando el algoritmo de N-Cuerpos, con las métricas de tiempo y velocidad de procesamiento
- d) **Optimización:** Escale el cluster utilizando el máximo de Raspberrys disponibles y obtenga la velocidad del clúster en GFLOPs

### Bibliografía:

- [1] Manual de instalacion de cluster Raspberry PI

## 6 Ordenamiento eficiente por ranking (Proyecto tipo A)

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



El proyecto debe contener lo siguiente:

- Elaborar un PRAM y el código en paralelo (MPI)
- Registro del desarrollo del código en por lo menos 3 pasos (versiones beta).
- Medir el tiempo de ejecución, y la velocidad del algoritmo, y representarlas en gráficas. Comparar los resultados con la complejidad teórica en paralelo, para distinto número de procesos y tamaño del problema. Utilice un tamaño adecuado del array para lograr tiempos de ejecución medibles
- Optimización:** Compare la escalabilidad del software con Quicksort en paralelo

### Bibliografía:

- [1] Curso Algoritmos Paralelos, Peter Sanders, 2020/2021

## 7 Indicaciones importantes para la entrega del proyecto

### 7.1 Informe de proyecto

Se pide realizar un informe de los resultados del proyecto, que contenga la siguiente información:

- **Nombre del proyecto**, y de los **integrantes**
- **Porcentaje de participación** de cada integrante en el proyecto (de 0 a 100 %, 100 es que ha contribuido en forma equitativa al resto)
- **Introducción:** Describa el problema y la solución planteada
- **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.

### 7.2 Entrega grupal

El proyecto se realizará en forma **grupal (2 o 3 integrantes)**

**El proyecto se subirá a Gradescope** (Proyecto Laboratorio)

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
Desarrollo de código/PRAM	El diseño del algoritmo y código es adecuado (6 pts)	Existen algunos errores menores en el diseño del algoritmo (3 pts)	Existen muchos errores en el diseño del algoritmo(0pts).
Optimización	Responde correctamente a la pregunta de optimización del resultado obtenido (4 pts)	El código contiene errores menores de programación. (2 pts).	El código contiene múltiples errores de programación (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).