# Método de Foster

**Ignacio Cortés Atzin Maxela 6IM2**

## Particionamiento

División del problema en tareas más pequeñas. Se identifica el máximo paralelismo potencial descomponiendo la computación y los datos asociados en pequeñas tareas.

**Descomposición funcional (*Functional Decomposition*)**: Divide las operaciones en tareas independientes (ej: etapas de un pipeline).

**Descomposición de datos (*Domain Decomposition*)**: Divide los datos en bloques (ej: matriz dividida en filas/columnas).

## Comunicación

Identificación de la comunicación necesaria entre las tareas definidas. Se establecen los canales de comunicación y se determina qué información debe intercambiarse.

Minimizar el **overhead** de comunicación y garantizar coherencia.

**Ejemplo**: En un filtro de imagen paralelo, las tareas vecinas deben intercambiar datos de bordes para evitar artefactos.

## Aglomeración

Combinación de tareas y comunicaciones para mejorar el rendimiento. Se evalúa si algunas tareas pequeñas deben agruparse para reducir la sobrecarga de comunicación.

**Ejemplo**: Agrupar múltiples operaciones de un bucle en una sola tarea para reducir comunicación.

## Mapeo

Asignación de tareas a procesadores específicos. Se busca balancear la carga de trabajo y minimizar la comunicación entre procesadores.

**Estático**: Asignación fija (ej: round-robin).

**Dinámico**: Balanceo en tiempo de ejecución (ej: work-stealing).

Una posible aplicación del cómputo paralelo en un **árbol de decisión** con Python consiste en paralelizar tareas como la **evaluación de atributos para dividir los datos** o la **construcción simultánea de subárboles izquierdo y derecho**, ya que estas operaciones son independientes. Utilizando la metodología del **método de Foster**, se puede dividir el problema en partes (atributos o subárboles), asignarlas a diferentes procesos (por ejemplo, con la librería multiprocessing), minimizar la comunicación entre ellos y mapear eficientemente las tareas a los núcleos disponibles. Esto mejora el rendimiento y permite entrenar árboles más rápido, especialmente en conjuntos de datos grandes o al construir múltiples árboles como en un Random Forest.

El método de Foster permite diseñar una versión paralela eficiente de un árbol de decisión en Python, especialmente útil en datasets con miles de características o muestras. Aunque la construcción del árbol tiene pasos secuenciales (como la división recursiva), la evaluación de splits y el entrenamiento de múltiples árboles (en ensambles) pueden paralelizarse siguiendo las etapas de **PCAM**. Herramientas como joblib, Dask o PySpark son clave para implementar esta metodología.

Referencias:

https://rinacional.tecnm.mx/bitstream/TecNM/3087/1/G10070519_donacion_tesis_bib.pdf