# Práctica 7: MPI básico

## Ejercicio 1: comunicación punto a punto bloqueante

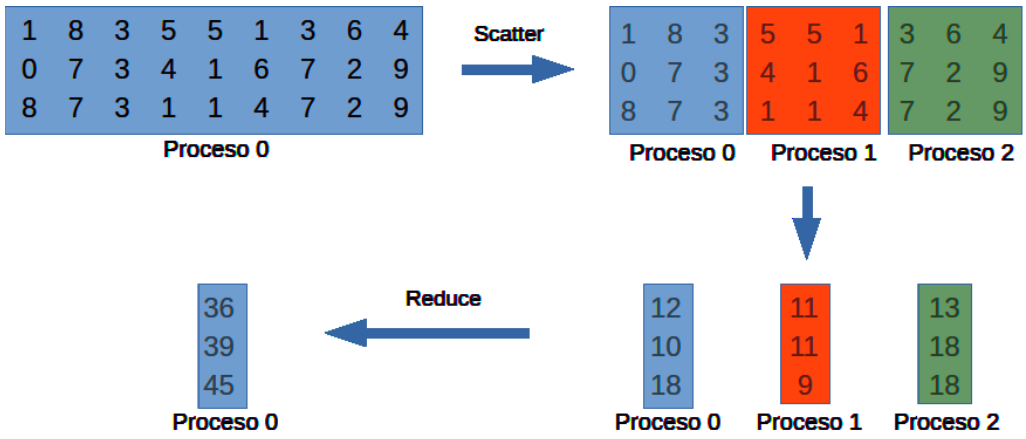
- Ejecutar el programa **01-StandardSendRecv.py** cambiando el número de procesos. Observar la salida
- Eliminar en el código el uso de los argumentos **tag** y **source** en la función **comm.Recv()**. Probar de nuevo
- Para obtener **tag** y **source** en recepción, antes de la recepción se debe ejecutar

  ```
  status = MPI.Status()
  ```

- y usar el argumento adicional **status=status** en **Recv**. A continuación se pueden usar las funciones **status.Get_tag()** y **status.Get_source()**. Probar el funcionamiento de esta manera
- Modificar el programa de tal forma que el proceso que recibe los mensajes sea el de rango p-1
- Modificar el programa inicial de tal forma que cada proceso de rango i mande un mensaje al proceso de rango i+1 (el último debe mandar el mensaje al proceso de rango 0). No usar el argumento **source** en recepción. ¿Qué pasa si el programa se ejecuta solamente con un proceso? ¿Estaremos en envío síncrono o tamponado?

## Ejercicio 2: comunicación colectiva

Partiendo del código **ComColectiva_TODO.py**, implementar en paralelo la suma de las columnas de una matriz A (L filas, L*NP columnas, con NP procesos)



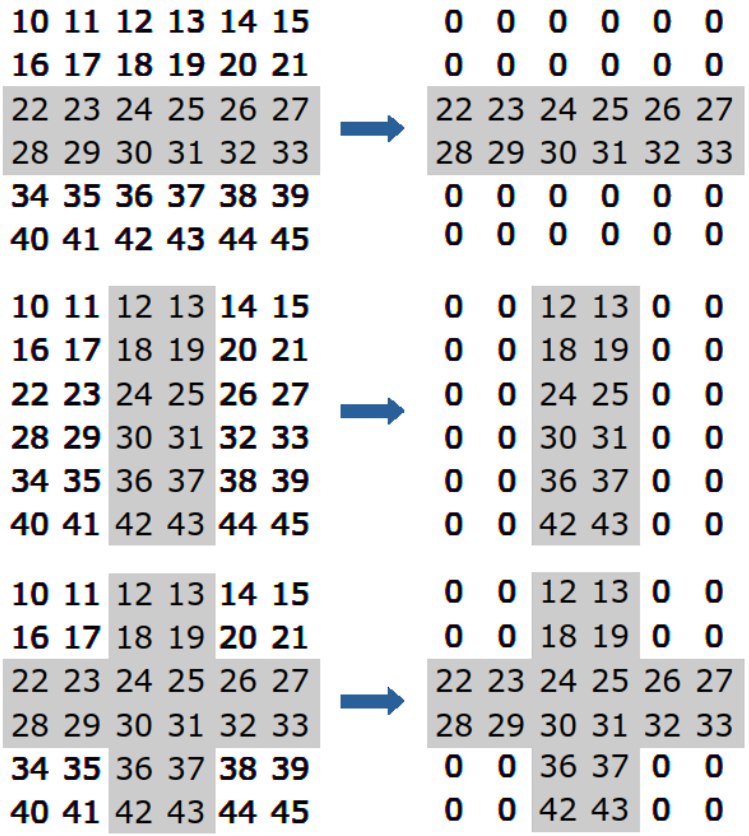

Resultado de la ejecución del programa:

(mpi) $ mpirun -np 3 ./ComColectiva.py

```
Matriz A:
[[5 7 3 6 1 4 0 4 4]
 [3 5 6 4 6 2 2 8 8]
 [4 3 6 2 0 0 5 7 4]]
Resultado:
[[34]
 [44]
 [31]]
(mpi) $
```

**Ejercicio 3: tipos derivados**

Usando tipos derivados, implementar un programa que cree y utilice los patrones de comunicación que se muestran




**Ejercicio 4: satélites geoestacionarios**

Se propone la implementación de una simulación de un sistema de objetos en el vacío que interaccionan gravitatoriamente dos a dos.

La ley de Newton de la gravitación universal establece que dos cuerpos de masa $m_1$ y $m_2$ separados una distancia d se atraerán con una fuerza dada por:

$$F = G\frac{m_1 m_2}{d^2}$$

siendo G la constante de gravitación universal

$$G = 6.674 \times 10^{-11}\frac{N\,m^2}{kg^2}$$

Un cuerpo de masa m que experimenta una fuerza F experimentará una aceleración

$$a = \frac{F}{m}$$

Se incluye un programa a completar que implementa de manera serie la simulación de un sistema de objetos de este tipo. Por motivos de simplicidad, se ha supuesto que el espacio es bidimensional, por lo que los vectores de posición, velocidad y aceleración de cada objeto estarán compuestos por dos valores. En un entorno de este tipo, la situación de cada objeto estará

definida por cinco valores: posición en x, posición en y, velocidad en x, velocidad en y, y masa. Obviamente, de éstos valores solo el último es constante.

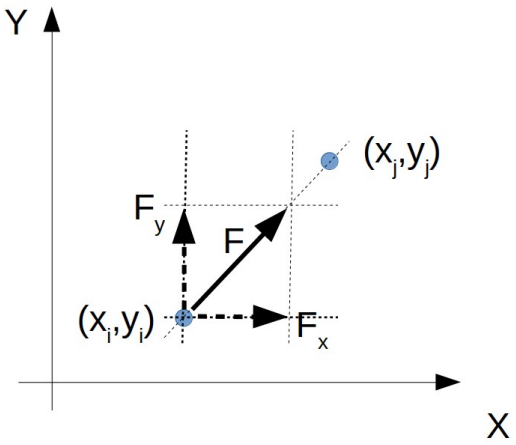
El funcionamiento del programa es iterativo, y para cada iteración realiza los siguientes pasos:

```
Para cada objeto i
   Para cada objeto j (i!=j)
      Calcular la distancia entre los objetos
      Calcular la fuerza gravitatoria sobre el objeto i realizada por j
      Calcular la contribución de la aceleración sobre i realizada por j
      Acumular las proyecciones sobre los ejes x e y de dicha aceleración
   Modificar la velocidad del objeto i
   Modificar la posición del objeto i
```

Las proyecciones de F sobre los ejes x e y ($F_x$ y $F_y$) vendrán dadas por las siguientes fórmulas:

$$F_x = F\frac{x_j - x_i}{d}$$

$$F_y = F\frac{y_j - y_i}{d}$$




El estado inicial del sistema viene definido en un fichero de texto, en el cual la primera línea es el número de objetos n, y las siguientes 5*n líneas darán, para cada objeto, su posición (x,y), su vector de velocidad ($v_x$, $v_y$), y su masa. El fichero **data4sat.txt** contiene la información correspondiente a la tierra y cuatro satélites geoestacionarios de 50 kg de peso cada uno.

### Apartado A

Para facilitar la implementación, se proporciona un esqueleto del programa en versión serie en el cual se han eliminado porciones de código que están marcadas con comentarios TODO. Se deben completar estas secciones de código para lograr la ejecución del programa.

### Apartado B

Partiendo del programa anterior, se realizará una implementación en MPI en el cual cada proceso se encargará de un objeto. Para ello, el proceso 0 leerá todos los datos del fichero y repartirá los objetos a los distintos procesos.

En cada iteración, cada proceso recibirá los datos de los restantes procesos y actualizará los datos de su objeto calculando las interacciones igual que en la implementación serie.

**Apartado C**

Partiendo de la implementación anterior, se pide incorporar al programa las siguientes funcionalidades:

- Implementar mediante comunicación colectiva las transmisiones de datos entre procesos.
- Implementar mediante tipos de datos derivados o empaquetamiento de datos la transmisión de datos entre procesos.

El objetivo es lograr que las comunicaciones se realicen con una sola línea. Por ejemplo, que el proceso 0 envíe sus datos (x, y, $v_x$, $v_y$, m) a los procesos 1, 2, … n-1 con una sola orden, y que el proceso 0 reciba todos los datos de los demás procesos con una sola orden.