

The background is a dark blue-grey gradient. On the left, there are two overlapping geometric shapes: a blue parallelogram and a light green parallelogram. Below these, a circular inset shows a detailed, high-contrast image of a computer circuit board. In the top right corner, there is a faint, stylized pattern of white lines resembling a circuit or a data flow diagram.

# Práctica MPI

## Arquitectura de Computadores

Juan Gil Sancho  
Pablo Caño Pascual  
Manuel García Galante  
Javier Galante Gómez

# Índice

[Introducción](#)

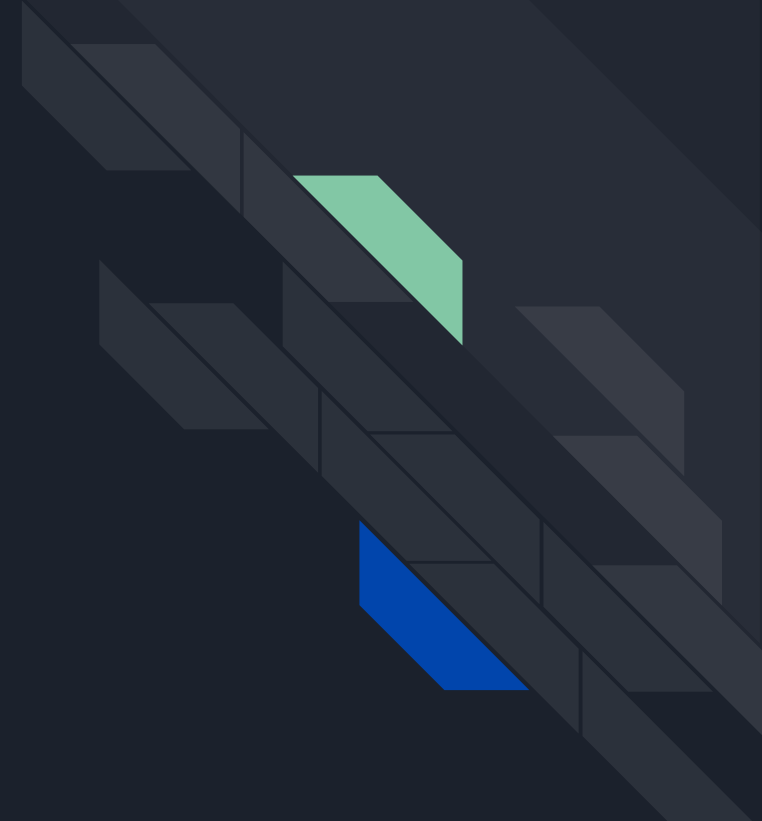
[Enfoque de la práctica](#)

[Objetivo de la práctica](#)

[Procesos](#)


[Gráficos](#)

[Conclusiones](#)



# Introducción

Para el estudio de eficiencia de nuestro algoritmo MPI, hemos realizado las pruebas en un sistema operativo basado en GNU linux en esta máquina:

Intel Core i7-1065G7 @ 1.30GHz		Average CPU Mark
Description: Intel Iris Plus Graphics		 <b>8719</b> Single Thread Rating: 2410 Samples: 1590* *Margin for error: Low <a href="#">+ COMPARE</a>  PerformanceTest V9 CPU Mark: 10,488 Thread: 2,519
Class: Laptop	Socket: FCBGA-1526	
Clockspeed: 1.3 GHZ	Turbo Speed: 3.9 GHZ	
Cores: 4 Threads: 8	Typical TDP: 15 W	
TDP Down: 12 W	TDP Up: 25 W	
Other names: Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz		
CPU First Seen on Charts: Q2 2019		
CPUMark/\$Price: 20.47		

Como vemos disponemos de 4 Núcleos y 8 hilos en un procesador Intel de décima generación basado en la arquitectura Ice Lake



# Enfoque de la práctica

01

Como proceso principal tenemos un proceso 0 que se comunica con los procesos generadores y comprobadores y es el encargado de la entrada y salida de datos

02

Los procesos comprobadores se encargan de comprobar las cadenas generadas por los generadores con la cadena secreta que le es enviada por el proceso 0 de E/S. El número de procesos comprobadores viene dado por el primer argumento que se le pasa al programa al ejecutarlo.

03

Los procesos generadores son los que generan cadenas aleatorias que serán comprobadas por los comprobadores. Reciben de estos una cadena pista por la que se guían al generar nuevas cadenas. El número de generadores depende del número de procesos y el número de comprobadores



# Objetivo de la práctica

El objetivo de esta práctica es observar cómo se reducen los tiempos de ejecución cuanto mayor cantidad de procesos que estemos utilizando en el modo pista (que es el modo en el que los procesos se comunican entre sí) mientras que en el modo no pista no habría una reducción en los tiempos tan importante al no comunicarse los comprobadores entre sí pasándose la pista que han encontrado al global de generadores



# Proceso de E/S

Es el proceso principal y se comunica con comprobadores y generadores

Envía a los comprobadores un flag para que inicie, el tamaño de la palabra y la palabra secreta.

Envía a los generadores un flag para que inicie, el id de su comprobador y el tamaño de la palabra que tiene que generar

```
/*Para comprobadores:
-> Les notificamos que son comprobadores
-> Les enviamos el tamaño de la cadena a adivinar
-> Les enviamos la cadena*/
flagTipo = 1;
for (i = 0; i < numComprobadores; i++)
{
    MPI_Send(&flagTipo, 1, MPI_INT, listaComprobadores[i], TIPO_PROC, MPI_COMM_WORLD);
    MPI_Send(&tamanoPalabra, 1, MPI_INT, listaComprobadores[i], 1, MPI_COMM_WORLD);
    MPI_Send(secret, tamanoPalabra, MPI_CHAR, listaComprobadores[i], 2, MPI_COMM_WORLD);

    // Muestra por pantalla notificación de tipo
    printf("%d\n", listaComprobadores[i]);
    fflush(stdout);
}
```

```
/*Para generadores:
-> Les notificamos que son generadores
-> El idWorld de su comprobador correspondiente
-> Les enviamos el tamaño de la cadena a adivinar*/
flagTipo = 0;
for (int k = 1, i = 0; i < numGeneradores; i++, k++)
{
    tmpGenIni = mygettime();
    /*Cuando el contador k supera el número
    de comprobadores vuelve al principio*/
    if (k > numComprobadores)
    {
        k = 1;
    }

    MPI_Send(&flagTipo, 1, MPI_INT, listaGeneradores[i], TIPO_PROC, MPI_COMM_WORLD);

    printf("%d\n", listaGeneradores[i], k);
    fflush(stdout);

    MPI_Send(&k, 1, MPI_INT, listaGeneradores[i], 1, MPI_COMM_WORLD);
    MPI_Send(&tamanoPalabra, 1, MPI_INT, listaGeneradores[i], 2, MPI_COMM_WORLD);
    tmpGenFin = mygettime();
    tmpGen = tmpGenFin - tmpGenIni;
}
```



# Proceso de E/S

También se encarga de recibir las pistas que son creadas por los comprobadores además de enviarla a todos los generadores con el modo pista activado

Además, se encarga de mandar un flag para que los comprobadores y generadores finalicen al encontrarse la palabra secreta.

```
while (!flagSalir)
{
    MPI_Recv(pistaRecibida, tamanoPalabra, MPI_CHAR, MPI_ANY_SOURCE, PISTA_ES, MPI_COMM_WORLD, &status);
    pistaRecibida[tamanoPalabra] = '\0';
    if (modo_pista)
    {
        for (int i = numComprobadores + 1; i < numProcesos; i++)
        {
            if (i != status.MPI_SOURCE)
                MPI_Send(pistaRecibida, tamanoPalabra, MPI_CHAR, i, 15, MPI_COMM_WORLD);
        }
    }
}
```

```
if (strcmp(pistaRecibida, secreta) == 0)
{
    flagSalir = 1;
    printf("\n\n\n\n PALABRA ENCONTRADA POR %d\n\n\n", status.MPI_SOURCE);
    fflush(stdout);
    for (int i = 1; i < numProcesos; i++)
    {
        MPI_Send(&flagSalir, 1, MPI_INT, i, 6, MPI_COMM_WORLD);
        if (i <= numComprobadores)
        {
            MPI_Send(pistaRecibida, tamanoPalabra, MPI_CHAR, i, 3, MPI_COMM_WORLD);
            printf("\n%d Fin comprobadores", i);
            fflush(stdout);
        }
        else
        {
            printf("\n%d Fin generadores", i);
            fflush(stdout);
        }
    }
}
```



# Comprobadores

Los comprobadores reciben un flag mediante un mensaje no bloqueante para terminar la ejecución.

Recibe una cadena de uno de sus generadores.

Comprueba la cadena recibida y envía al generador una pista con los caracteres que ha acertado.

```
while (!flagSalir)
{
    MPI_Test(&peticionSalir, &flagSalir, &status);
    // Recibe en la cadena propuesta la cadena generada por el generador
    MPI_Recv(generada, tamanoPalabra, MPI_CHAR, MPI_ANY_SOURCE, 3, MPI_COMM_WORLD, &status);
    if (!flagSalir)
    {
        generada[tamanoPalabra] = '\0';

        // Comprueba la cadena y guarda en la cadena acertado los caracteres coincidentes
        tmpCompIni = mygettime();
        comprobarCadena(tamanoPalabra, secreta, generada, cadena_acertada);
        tmpCompFin = mygettime();

        // Envía la cadena al generador con los caracteres que ha acertado
        MPI_Send(cadena_acertada, tamanoPalabra, MPI_CHAR, status.MPI_SOURCE, 4, MPI_COMM_WORLD);
        tmpComp += tmpCompFin - tmpCompIni;
        iteraciones++;
    }
}
```





# Generadores

Los generadores fusionan su pista con la recibida desde su comprobador y generan una nueva palabra con respecto de esta nueva pista.

Envían la palabra generada al comprobador y reciben la pista que este les devuelve.

En caso que la pista devuelta por el comprobador sea igual a la palabra generada anteriormente, comunica al proceso de E/S que ha encontrado la palabra correcta.

```
while (!flagSalir)
{
    tmpGenIni = mygettime();

    if (modo_pista)
    {
        MPI_Test(&peticion_nuevapista, &flag_nuevapista, &status);
        if (flag_nuevapista)
        {
            fusionar_pistas(tamanoPalabra, c_pista, nueva_pista);
            MPI_Irecv(nueva_pista, tamanoPalabra, MPI_CHAR, 0, 15, MPI_COMM_WORLD, &peticion_nuevapista);
        }
    }
    generarPalabra(tamanoPalabra, palabra, c_pista);

    strcpy(c_anterior, c_pista);
    MPI_Send(palabra, tamanoPalabra, MPI_CHAR, miComprobador, 3, MPI_COMM_WORLD);
    tmpGenFin = mygettime();
    tmpGen += tmpGenFin - tmpGenIni;

    tmpEspCompIni = mygettime();

    MPI_Recv(c_pista, tamanoPalabra, MPI_CHAR, miComprobador, 4, MPI_COMM_WORLD, &status);
    tmpEspCompFin = mygettime();

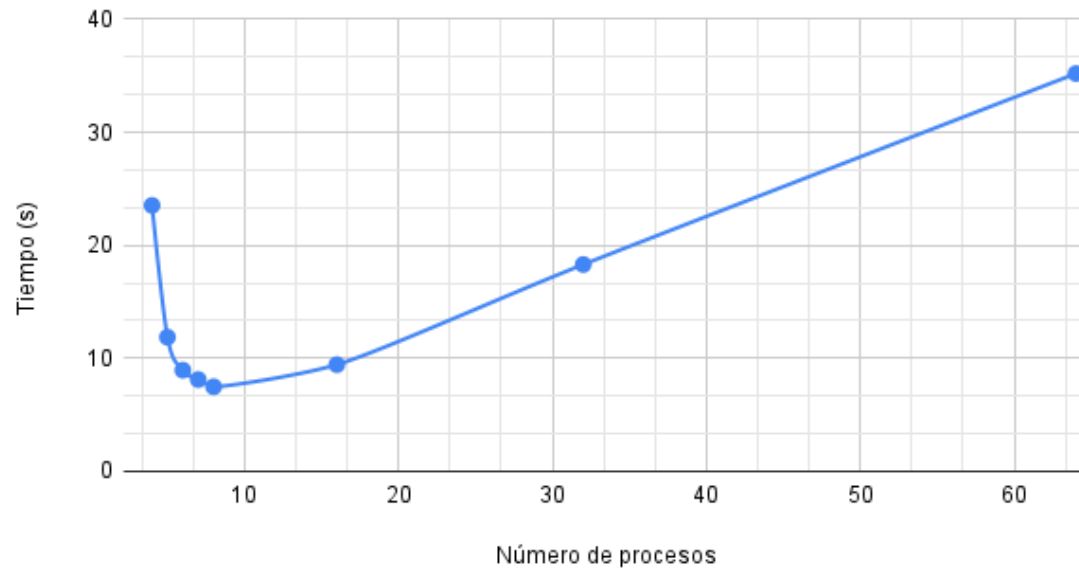
    tmpEspComp += tmpEspCompFin - tmpEspCompIni;

    if (strcmp(c_anterior, c_pista) != 0)
        MPI_Send(c_pista, tamanoPalabra, MPI_CHAR, 0, PISTA_ES, MPI_COMM_WORLD);

    iteraciones++;
}
```

# Gráfica 1 (Modo Pista)

Tiempo de ejecución respecto de número de procesos



## Gráfica 2 (Modo Pista)





# Modo pista vs modo sin pista

Para 8 procesos y 3 comprobadores, compararemos estos dos modos.

						TOTAL
Nº PROCESOS	COMPROBADORES	GENERADORES	PISTAS	COMPROBACIONES	TIEMPO	COMPROBACIONES/SEG
8	3	4	1	173	7,34	23,56948229
8	3	4	0	449	14,2	31,61971831



# Conclusiones

## Modo sin pista vs modo con pista:

- Al comparar el modo pista con el sin pista, es notable la diferencia de tiempo hasta averiguar la cadena secreta, pues el no comunicar los descubrimientos de caracteres entre los procesos hace que no sea una búsqueda eficiente.

## Gráfica 1:

- En el modo sin pista obtenemos una mejora de resultados a mayor número de núcleos hasta el límite de 8 de la máquina en la que trabajamos a partir del cual no existe mejora pues aunque MPI utilice más procesos, en realidad solo dispone de 8 procesadores por lo que el tiempo de ejecución no sólo no mejora, sino que empeora. Esto se debe a que al competir varios procesos por un mismo procesador hace que se pierda mucho tiempo cambiando de proceso en ejecución.

## Gráfica 2:

- Vemos que las comprobaciones por segundo aumentan hasta llegar a ocho procesos y luego se estabiliza en ese valor. También aumenta hasta dieciséis pero creemos que es una anomalía dada por no haber tomado más que una medida de cada en vez de varias y hacer una media.