

DESENCRYPTACIÓN POR FUERZA BRUTA

---

MPI

### ¿QUÉ ES MPI?

- ▶ Estándar utilizado para definir la sintaxis y semántica de las funciones de librerías de paso de mensajes, utilizadas para sacar partido a múltiples procesadores.
- ▶ El paso de mensajes al trabajar con sincronización y exclusión mutua es ideal para programación concurrente y sistemas distribuidos (no hace uso de memoria compartida).
- ▶ Elementos principales: proceso que envía, el que recibe y el mensaje

### LA PRÁCTICA

- ▶ El objetivo es implementar un algoritmo para descifrado de contraseñas con fuerza bruta haciendo uso de la librería MPI y observar como se desempeña en función del número de procesos y de máquinas.
- ▶ Ha dado como resultado un descriptador capaz de resolver 20 contraseñas de longitud 6 distribuyendo el trabajo entre los procesos que se crean. Posteriormente se han elaborado gráficas que permiten observar el rendimiento (Eje X: procesos, Eje Y: segundos o número de repeticiones de todos los procesos)

## PRIMEROS RESULTADOS

- ▶ Padre no participa en la tarea vs padre sí participa.
- ▶ Procesador: i5 4x2,6Ghz

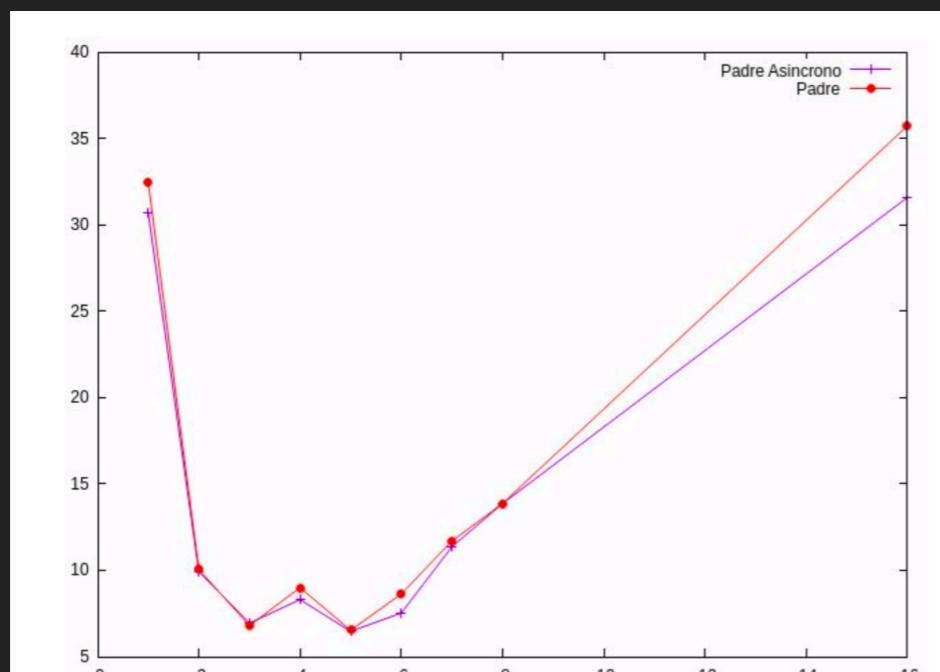


Figura 3. Tiempo(s)/nº procesos

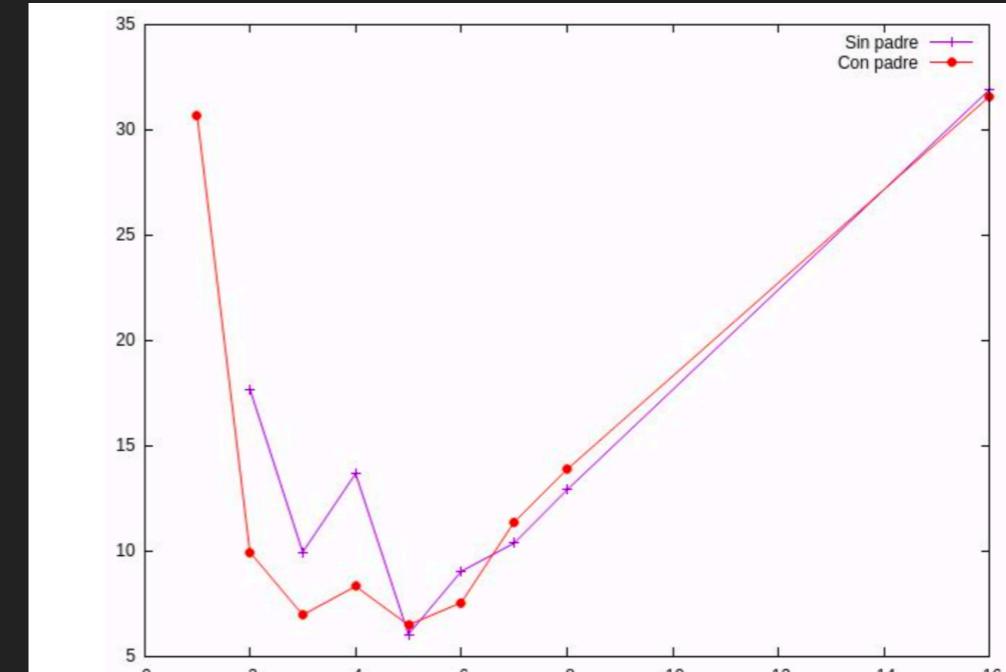


Figura 1. Tiempo(s)/nº procesos

## MEJORA 1

- Mejora: Paso de mensajes asíncrono, se evita el cuello de botella cuando el padre debe informar a cada proceso sucesivamente

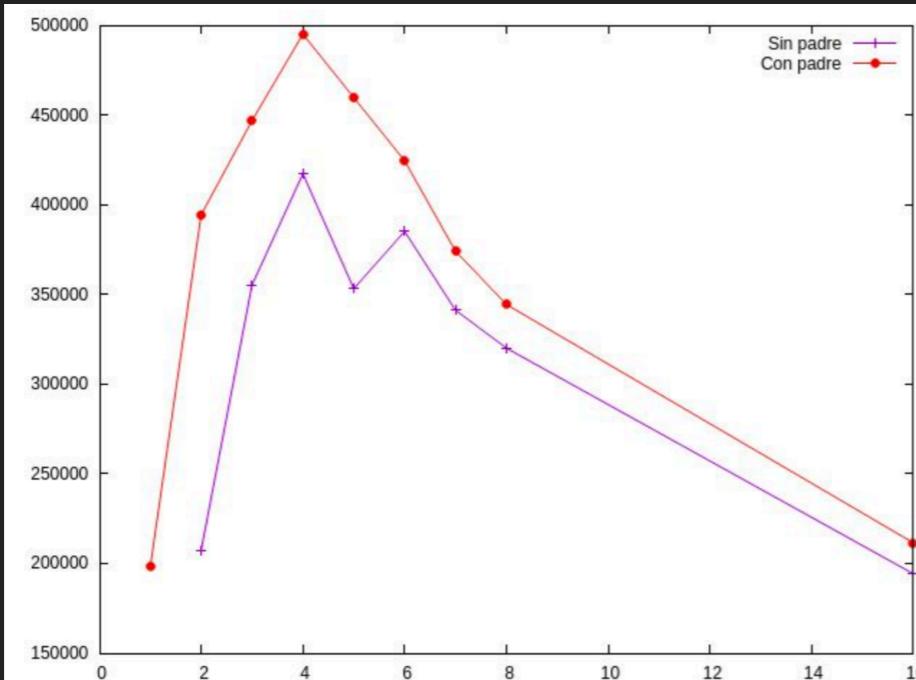


Figura 2. Repeticiones/nº procesos

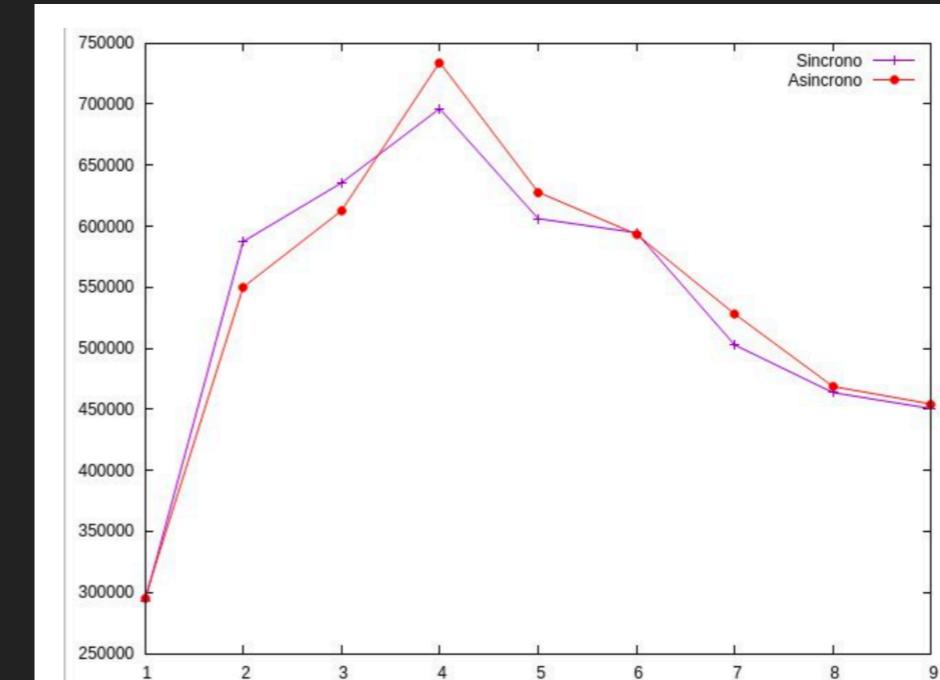


Figura 4. Repeticiones/nº procesos

## MEJORA 2

- ▶ Se ha modificado el intervalo de tiempo para llamar a la función MPI\_Iprobe() de comprobación de nuevos mensajes.

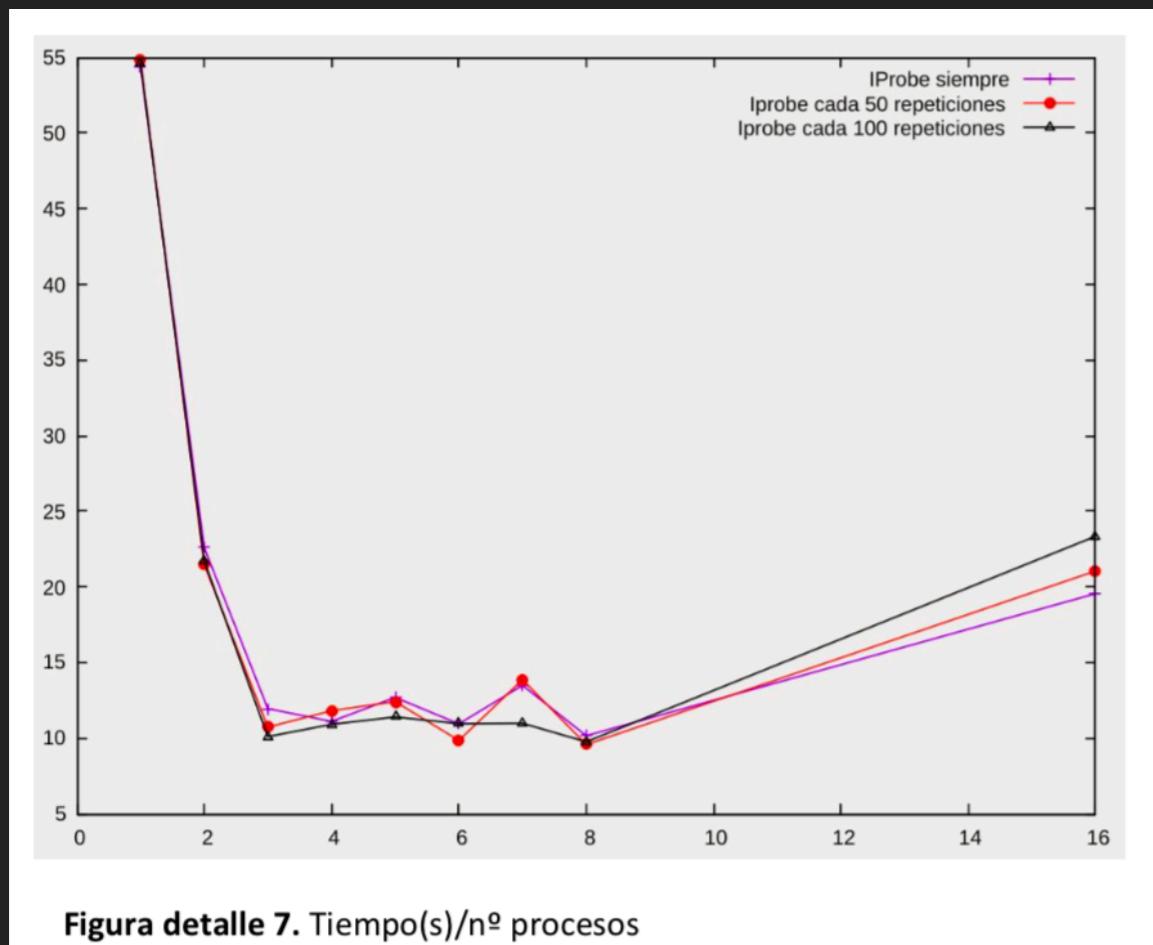


Figura detalle 7. Tiempo(s)/nº procesos

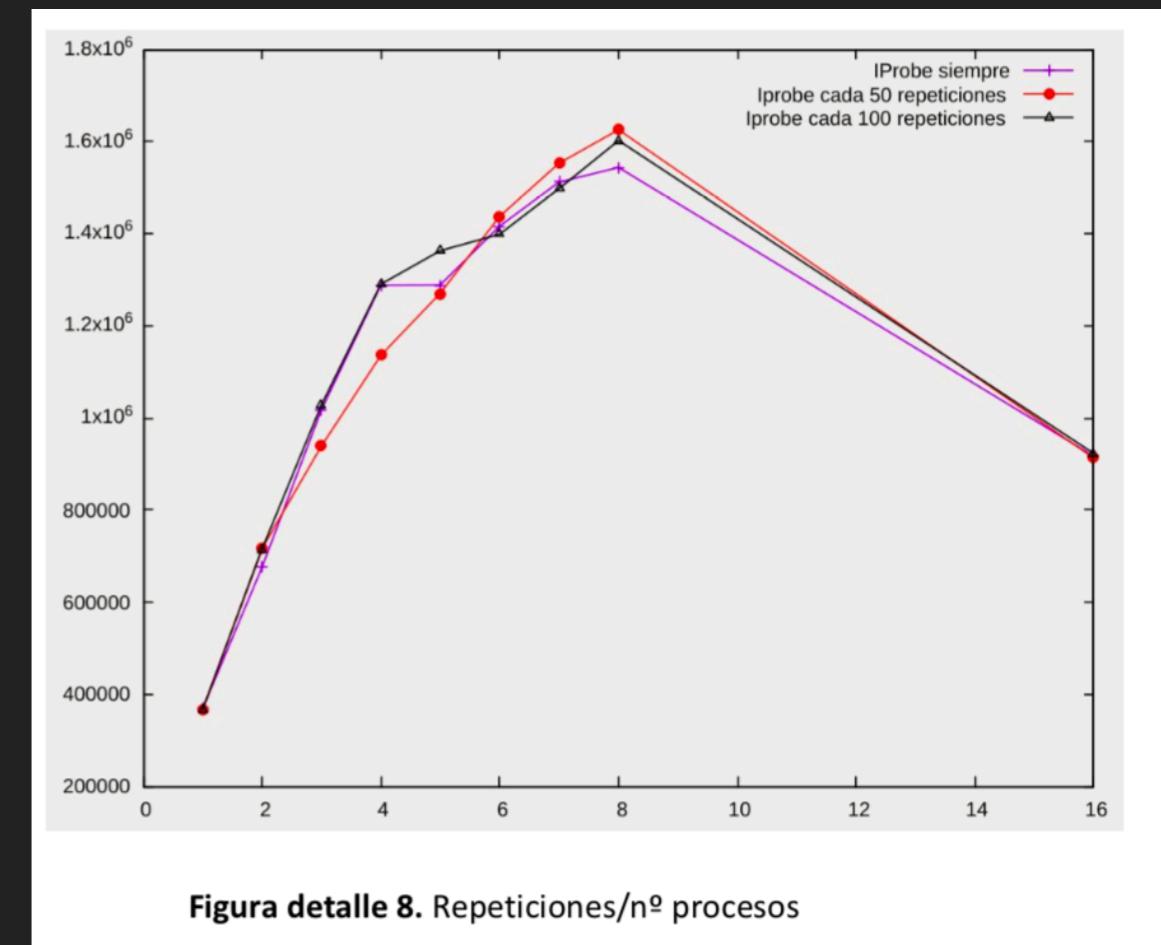


Figura detalle 8. Repeticiones/nº procesos

### MEJORA 2

- ▶ Como la mejora aportada no es lineal al aplicar ese cambio, se ha dejado la versión asíncrona con comprobación siempre de mensajes.

# RESULTADOS FINALES

- ▶ Procesador: i7 8x3,2GHz
- ▶ 20 contraseñas de longitud 6
- ▶ Los resultados finales salen de la media de 6 repeticiones

## RESULTADOS FINALES

- ▶ Cuando la CPU distribuye el trabajo el tiempo se reduce, siempre y cuando el numero de procesos no supere al de núcleos. De lo contrario la compartición de recursos anula el efecto conseguido por distribuir la carga de trabajo

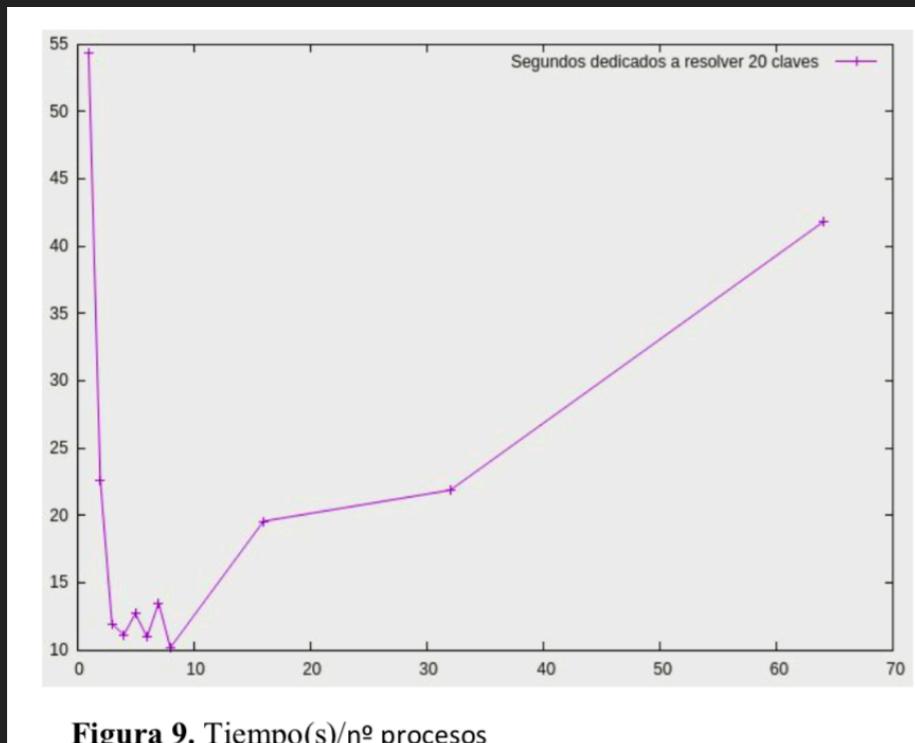


Figura 9. Tiempo(s)/nº procesos

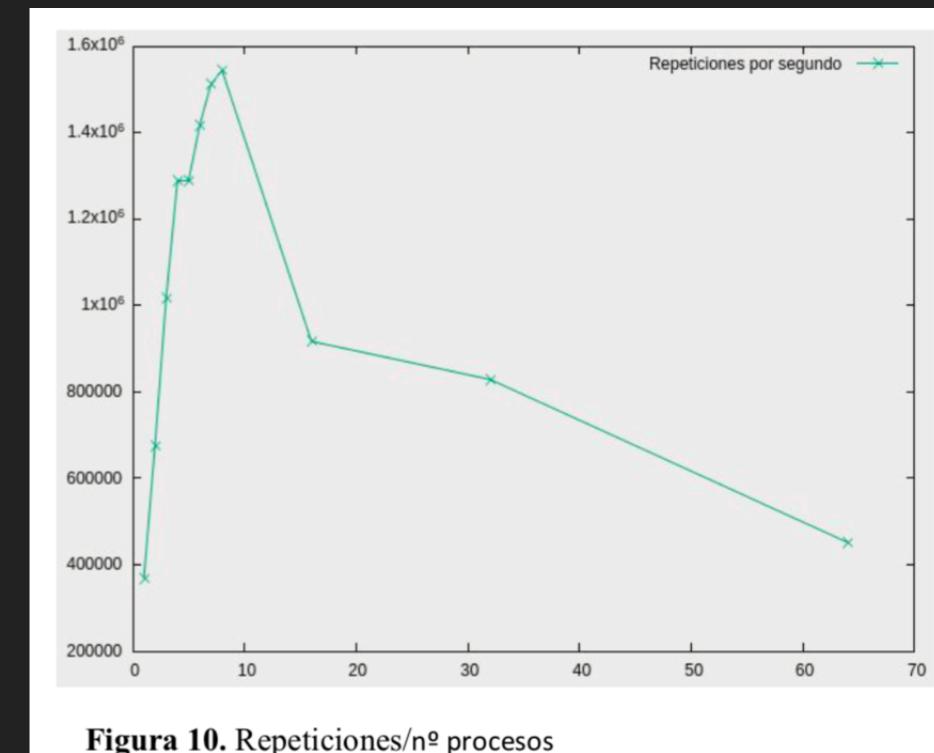


Figura 10. Repeticiones/nº procesos

## RESULTADOS FINALES

- Aumentando el número de máquinas interconectadas en 2 y 4 respecto a solo 1 se obtienen diferencias sustanciales hasta un límite.

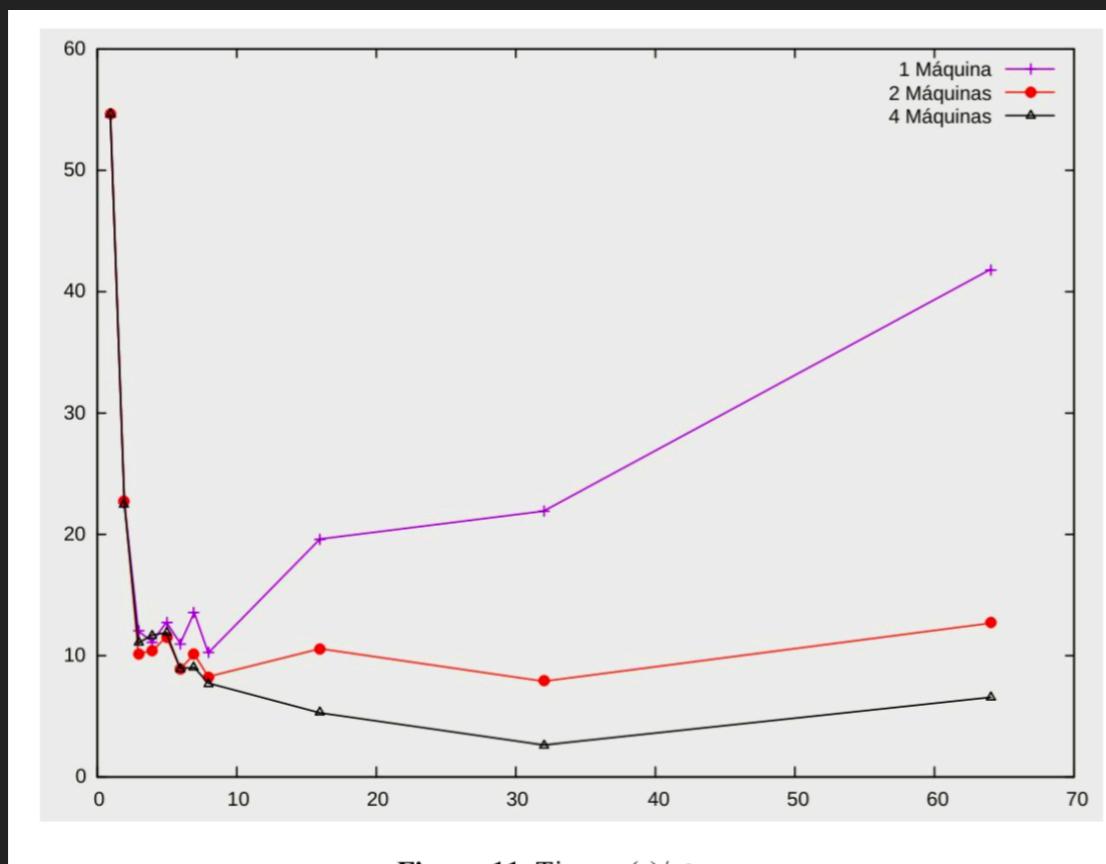


Figura 11. Tiempo(s)/nº procesos

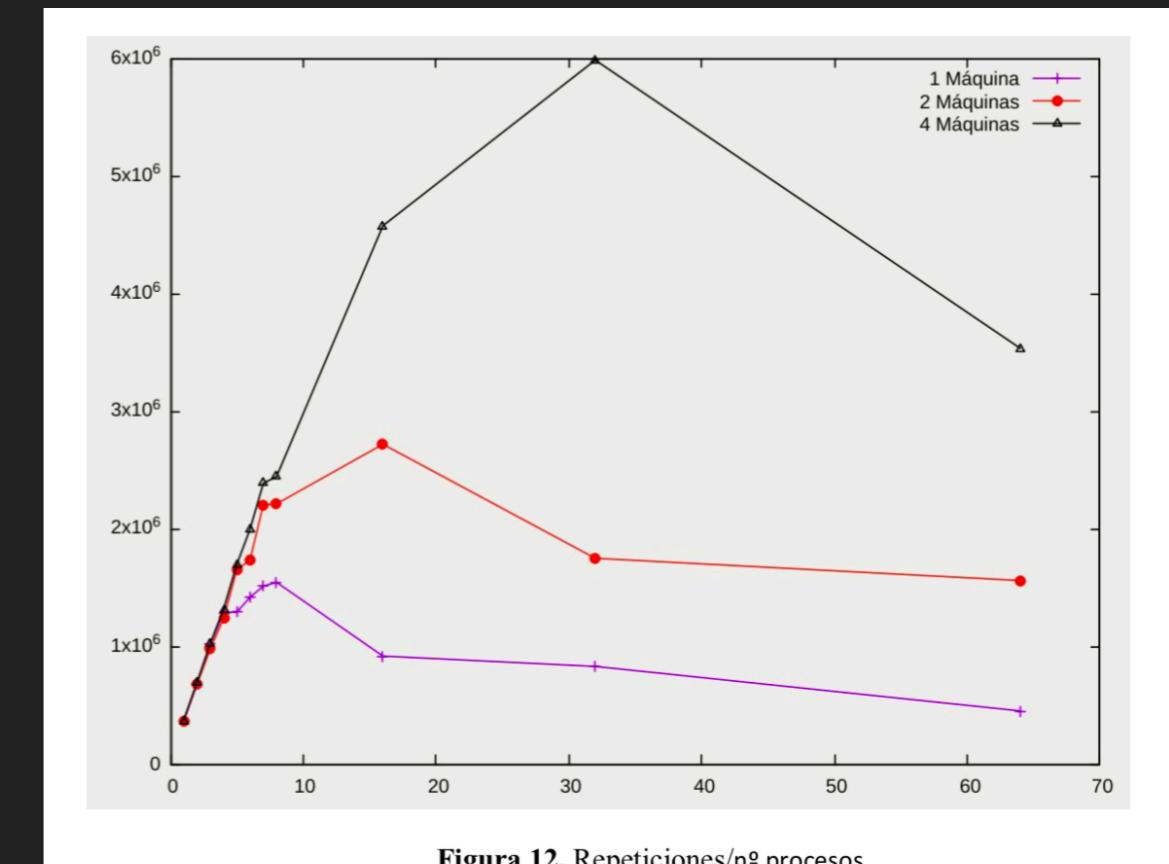


Figura 12. Repeticiones/nº procesos

## RESULTADOS FINALES

- Con 128 y 256 procesos el tiempo se dispara y empeora

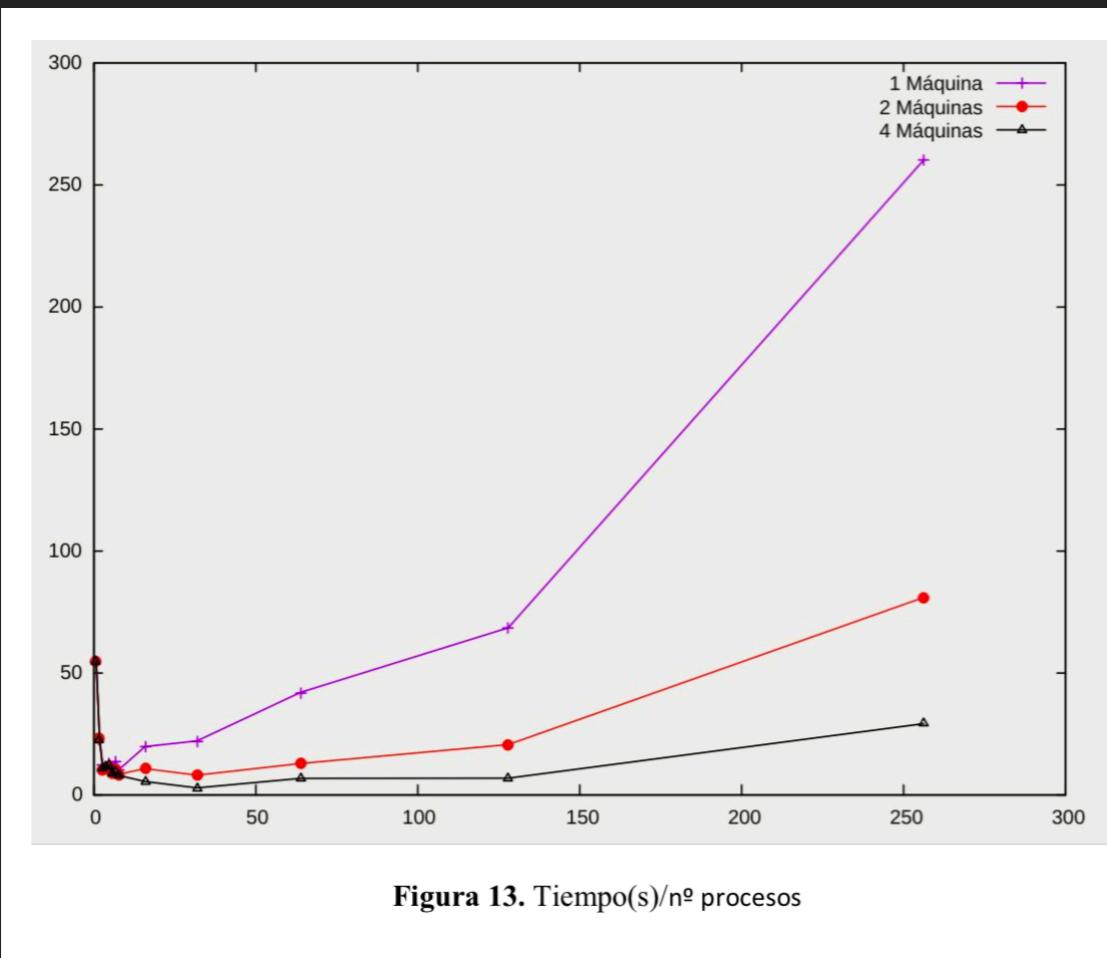


Figura 13. Tiempo(s)/nº procesos

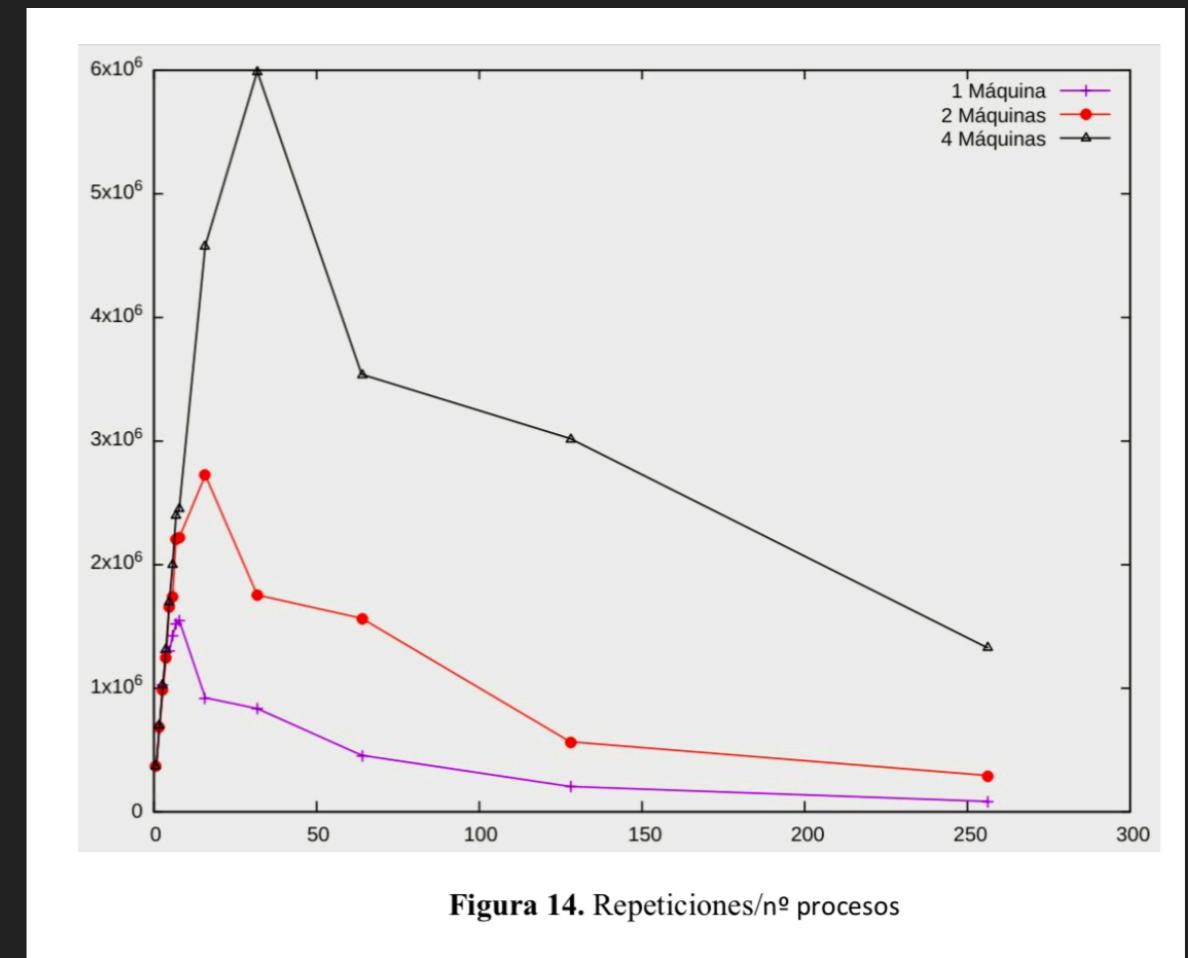


Figura 14. Repeticiones/nº procesos

## RESULTADOS FINALES

- Partiendo de que con un aumento de máquinas se obtienen diferencias notables, se han usado 6 y 8 también. En este caso no es tan apreciable aunque sí mejora.

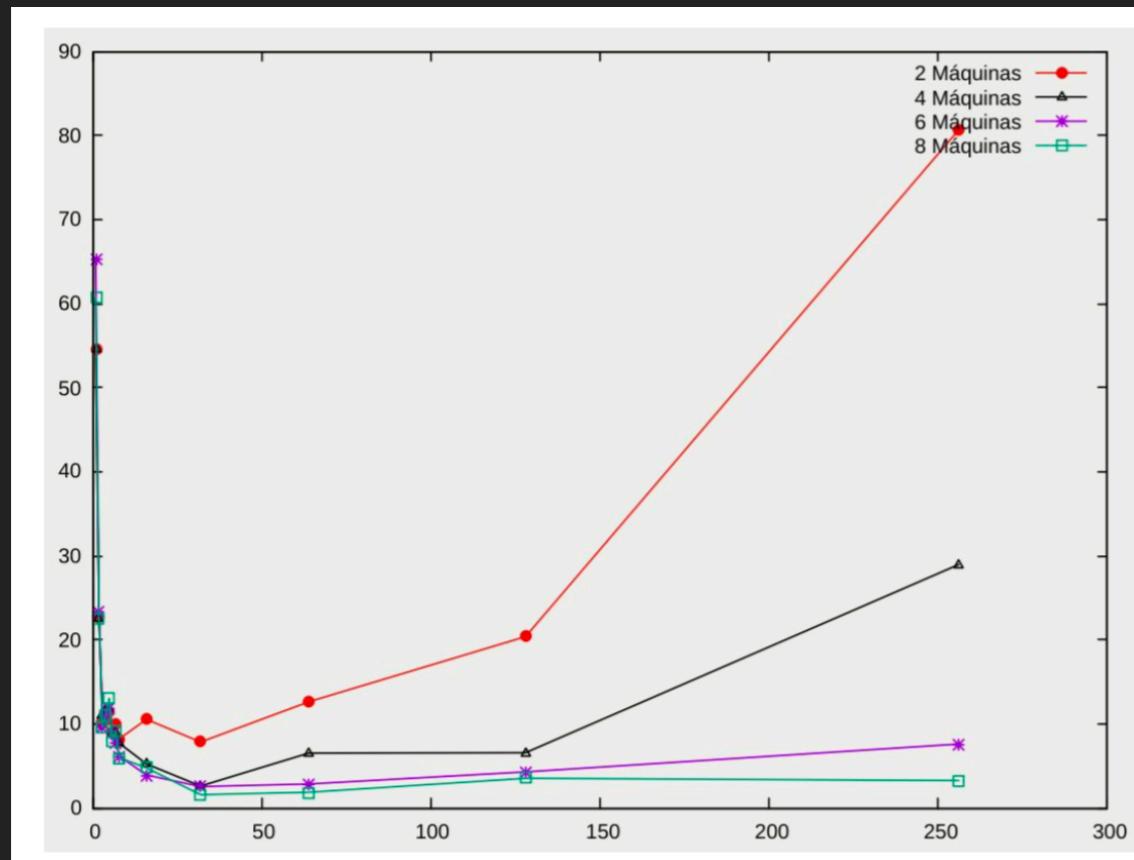


Figura 15. Tiempo(s)/nº procesos

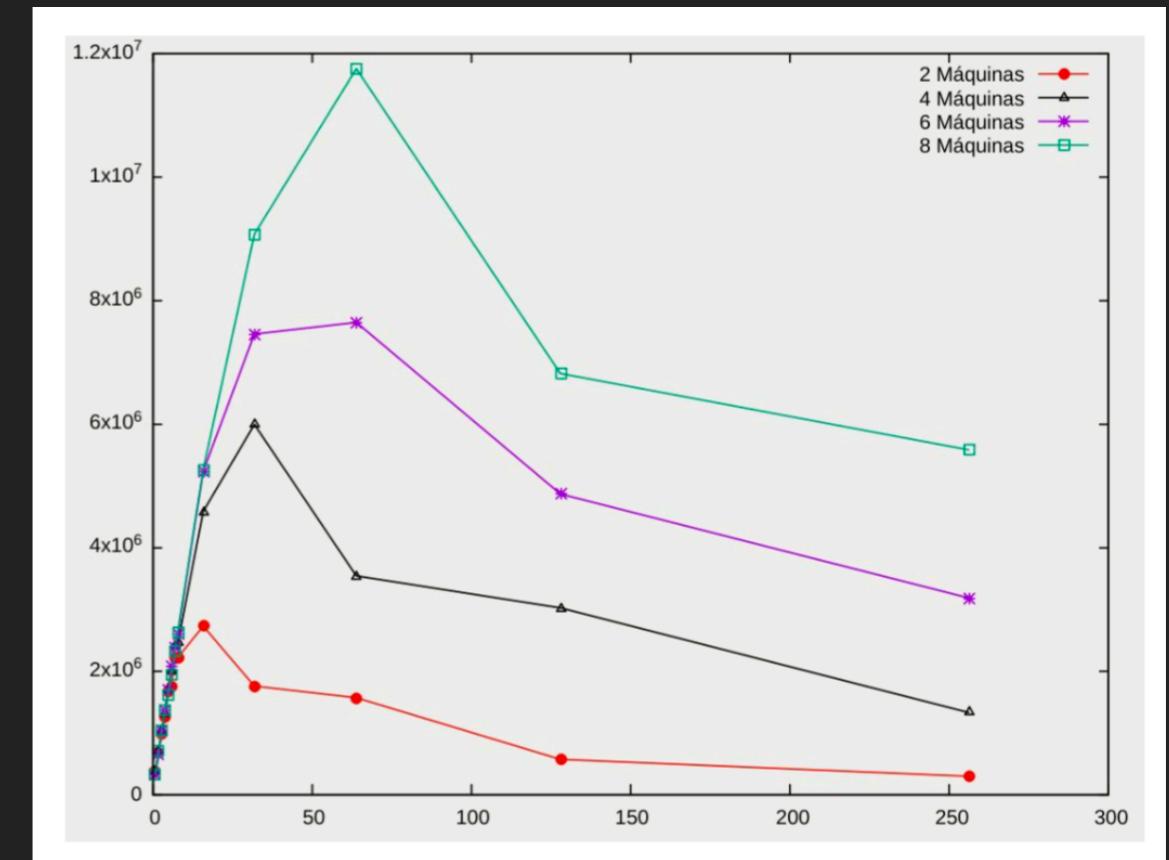


Figura 16. Repeticiones/nº procesos

## RESULTADOS FINALES

- ▶ Finalmente se resumió todo en una comparativa general
- ▶ La diferencia se produce a partir de 15 procesos y el punto óptimo de máquinas serían 6 en caso de decidir cuantos recursos económicos destinar. Con 8 no es notable.

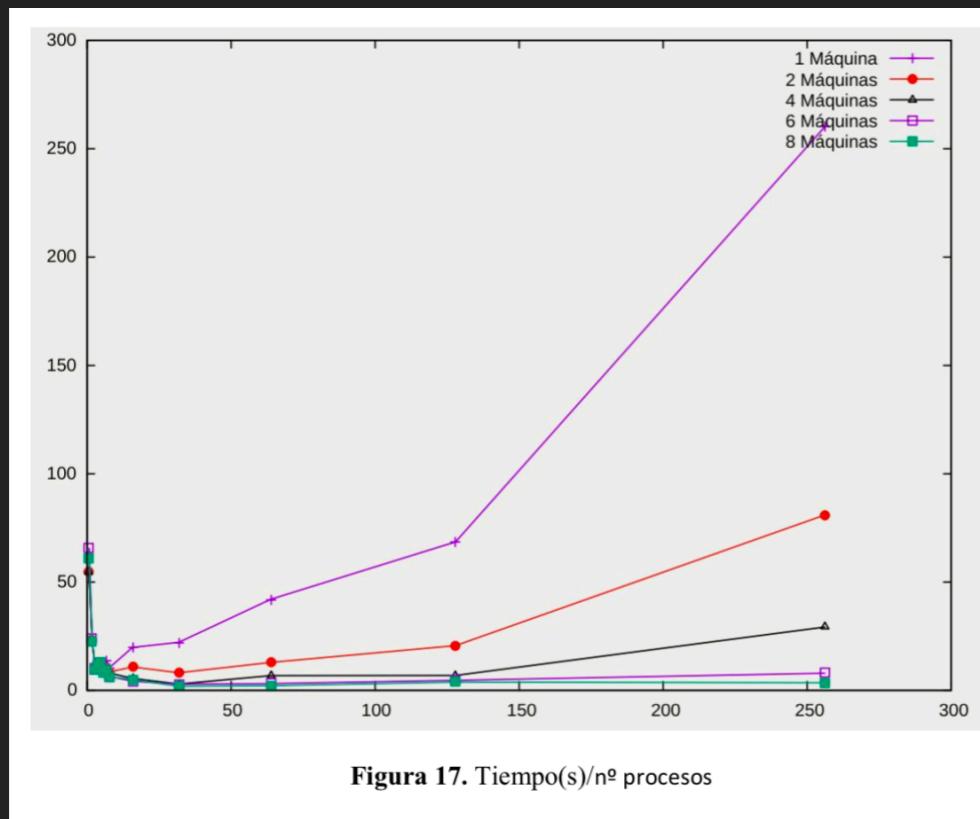


Figura 17. Tiempo(s)/nº procesos

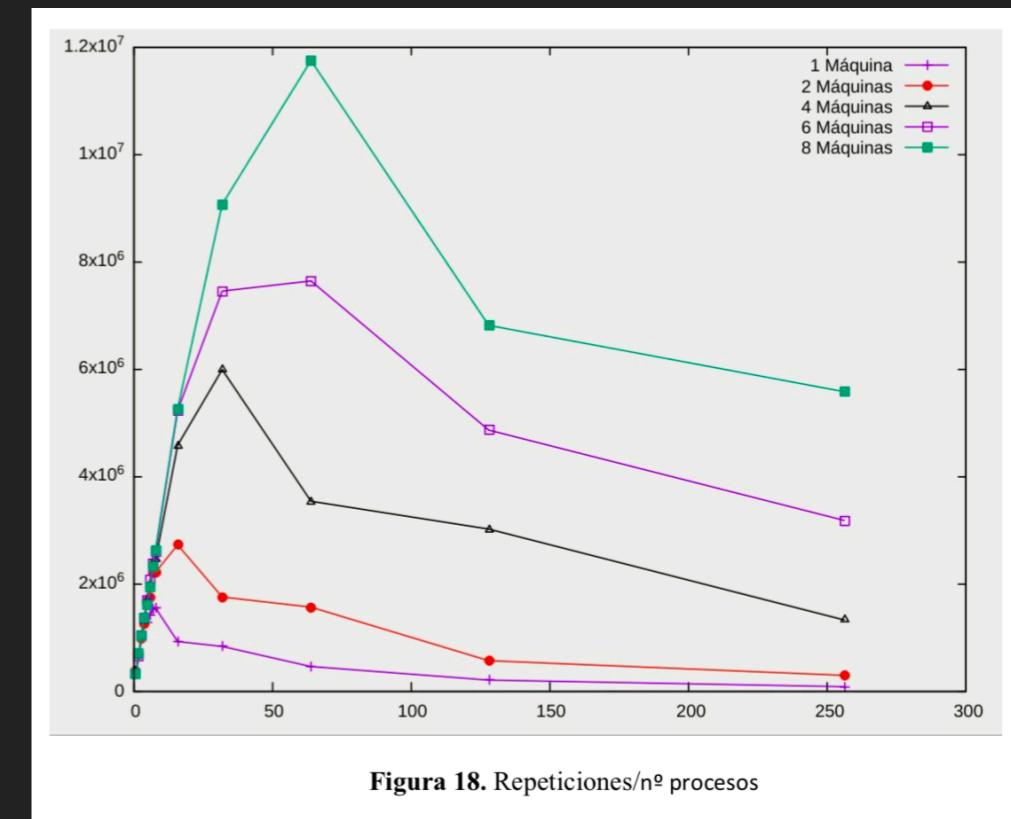


Figura 18. Repeticiones/nº procesos

## AUTORES

- ▶ Adrián Valera Román
- ▶ Jaime De La Peña Ramos
- ▶ Diego Mateos Matilla
- ▶ Javier Servate Hernández