ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

SISTEMAS OPERATIVOS

EVALUACIÓN DE RENDIMIENTO DE PROGRAMAS CON BASH REDUCE

**Integrantes:**

* Pedro Iñiguez
* Ray Montiel
* Roberto Yoncón

***Metodología***

Bash Reduce es una plataforma de procesamiento distribuida, que permite ejecutar grandes programas repartiendo la carga entre los núcleos que tenga la computadora.

En este caso nosotros procedimos a realizar 3 diferentes experimentos sobre conjuntos de datasets inmensos y algunos medianamente grandes.

El primero de ellos se trata sobre un parser porque se necesitan retirar ciertos caracteres para trabajar sobre los xml de manera cómoda.

El segundo es un encriptador sencillo, esto es necesario para enviar información oculta y que aunque esta sea intervenida, no pueda ser descifrada por los no destinatarios.

Por último tenemos un convertidor de mayúsculas a minúsculas, esto es útil para verificar que sólo exista un username en bases de datos que necesitan de un identificador para un usuario, para esto se puede convertir todos los caracteres del username a mayúsculas o minúsculas y compararlos con los ya existentes.

***Experimentos***

EXPERIMENTO ROBERTO YONCÓN

El experimento consiste en un convertidor de mayúsculas a minúsculas, el cual se encarga de revisar caracter a caracter si este es mayúscula o minúscula e invierte su estado, es decir las mayúsculas se harán minúsculas y viceversa.

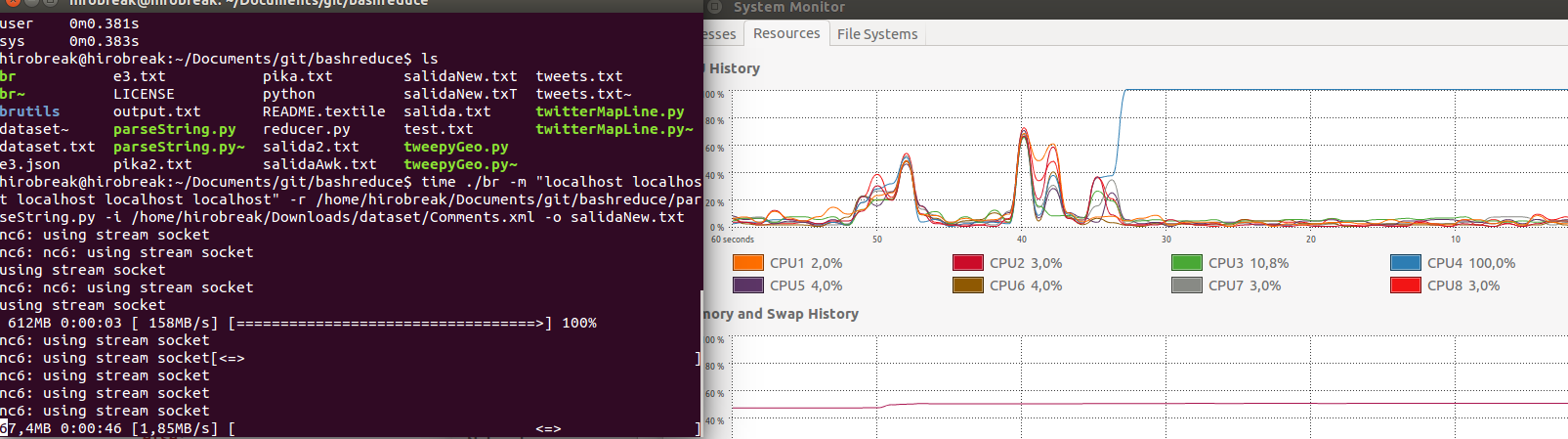
EXPERIMENTO PEDRO IÑIGUEZ

El experimento trata sobre un parser que se encarga de retirar todas las comillas del documento; revisa caracter a caracter si encuentra alguna y la retira del documento final, cuando encuentra algo diferente a las comillas revisa si es un dígito o un caracter conocido, en el caso de los dígitos los sumara y con respecto a los caracteres los agrega en una variable de frase.

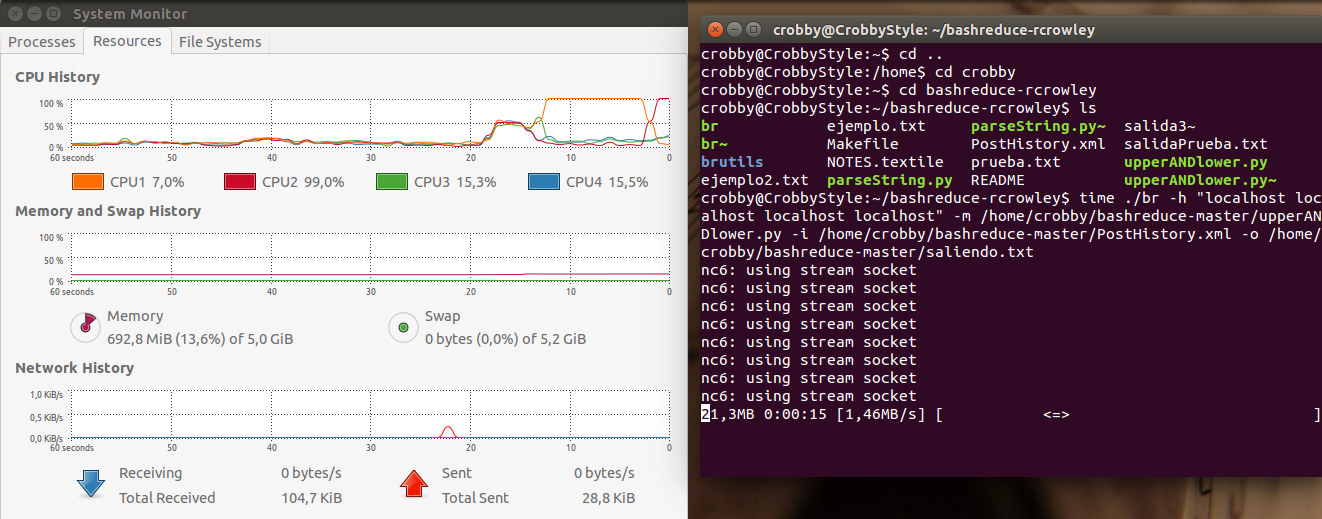
EXPERIMENTO RAY MONTIEL

El experimento es un encriptador sencillo, el cuál se encarga de leer caracter a caracter el dataset y luego aumenta en 3 su codificación ascii, aislando siempre los caracteres( :, “, \, #,etc).

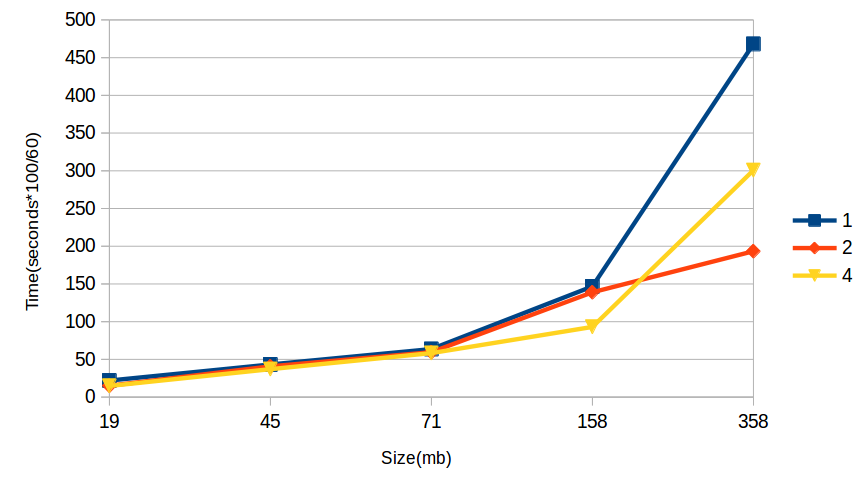
***Resultados***



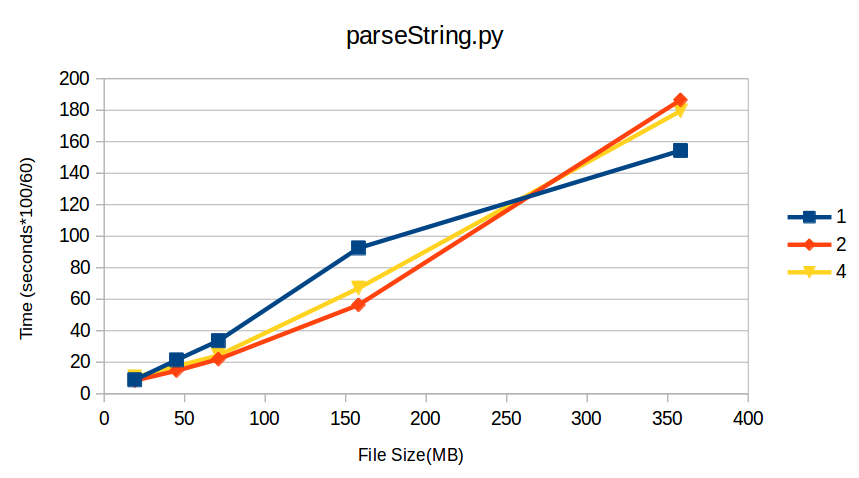
***Fig. 1.- Monitor del sistema del parseString***



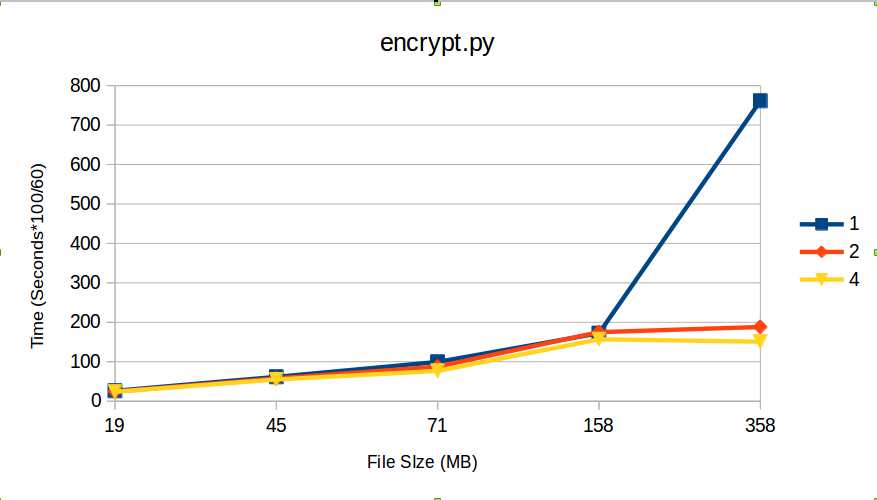
***Fig. 2.- Monitor del sistema del upperANDlower.py***



***Fig. 3.- Pruebas de upperANDlower.py script***



***Fig. 4.- Pruebas de parseString.py script***



***Fig. 5.- Pruebas de encrypt.py script***

***Conclusiones/ Análisis***

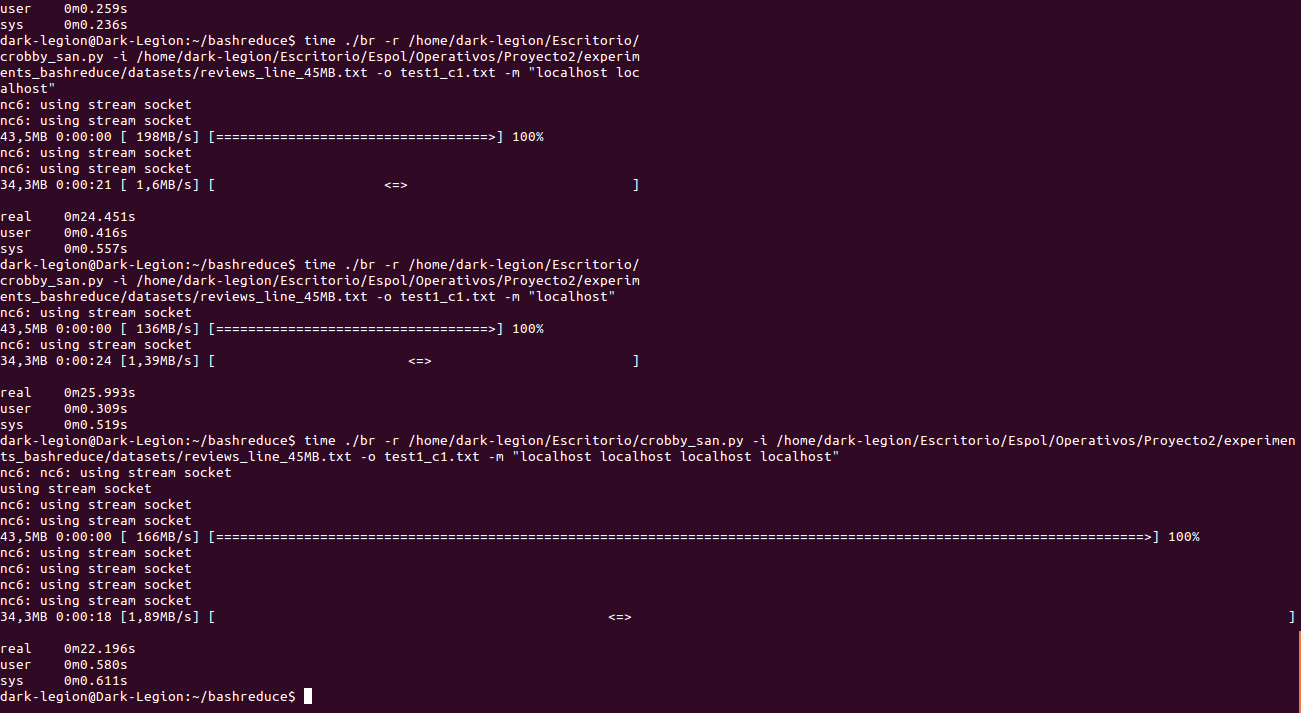
En la Fig.3 se aprecian los detalles del script upperANDlower el cual transforma de mayusculas a minusculas y viceversa cada caracter, en azul se aprecia la curva con 1 core el cuál llega aproximadamente a un valor de 500, con 2 cores se nota una mejora de aproximadamente 56% debido a que se reparte el procesamiento entre los núcleos y se pasa de mayúsculas a minúsculas mucho más rápido. Mientras que con 4 cores sólo mejora un 35%. El aumento de 2 a 4 cores se debe a que se realizó el experimento en una máquina de tan sólo 2 cores.

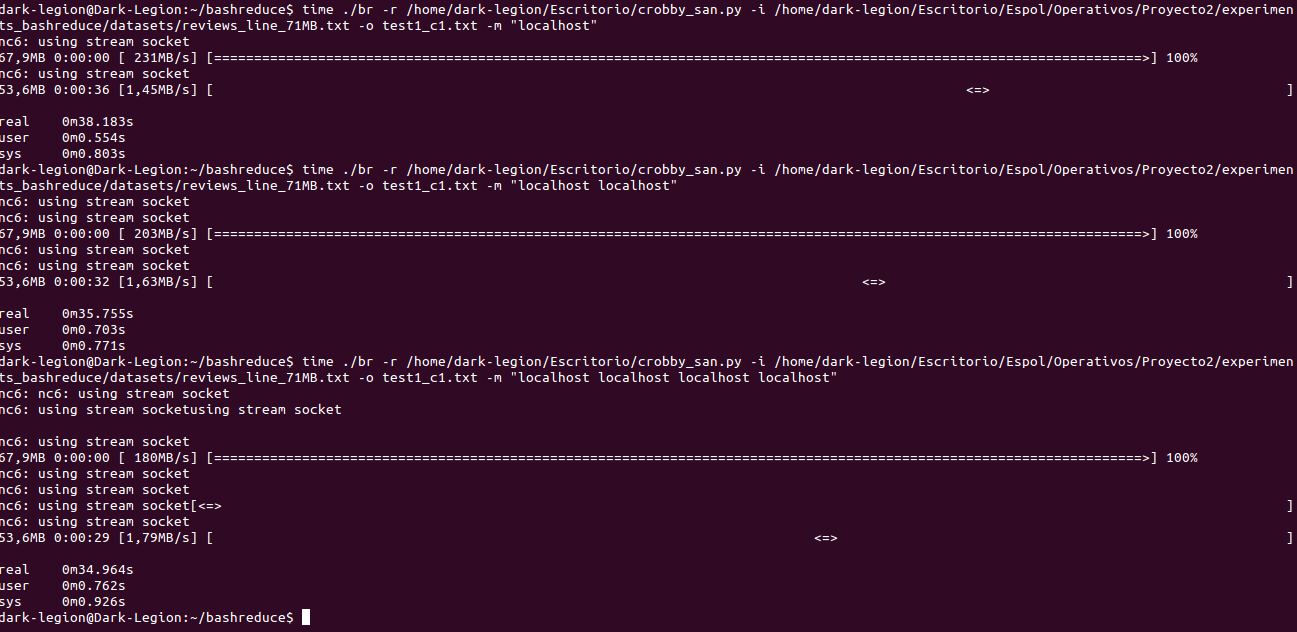
En la Fig.4 se aprecian los detalles del script parseString.py el cuál limpia de comillas al documento. En azul se visualiza que la curva con 1 core, el cual demora más que los demás excepto en el dataset de 358mb. Esto se debe a que posiblemente en aquel dataset no existían muchas comillas presentes.

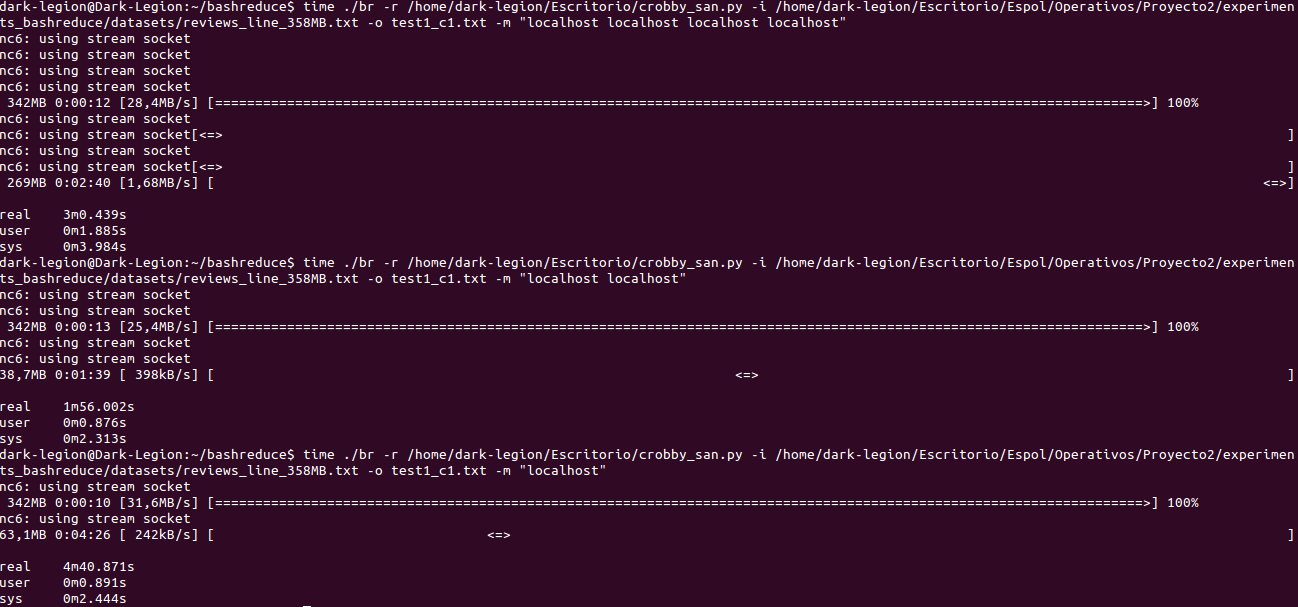
En la Fig.5 se aprecian los detalles del script encrypt.py el cuál obtiene cada caracter y le agrega 3 en su codificacion ascii, es decir si obtiene una ‘a’, devuelve una ‘d’. En azul se observa la curva de 1 core, el cual siempre es mayor en cada dataset, con 2 y 4 cores se observa una notable mejora porque el sistema reparte el procesamiento entre todos los núcleos.

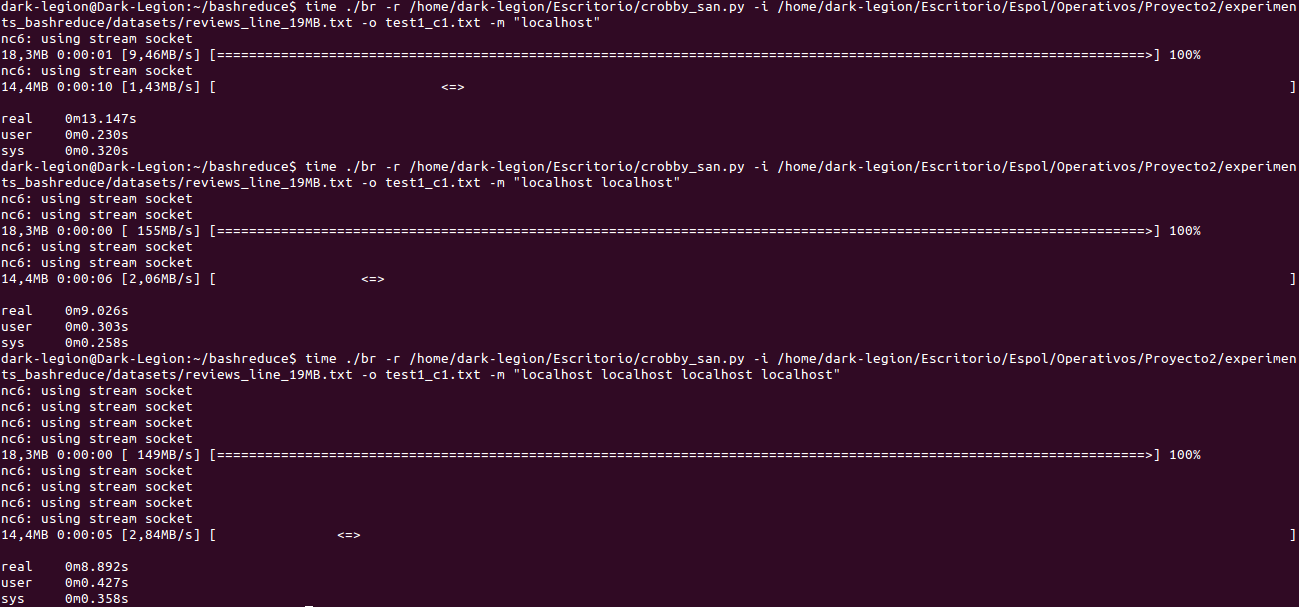
***Anexos***

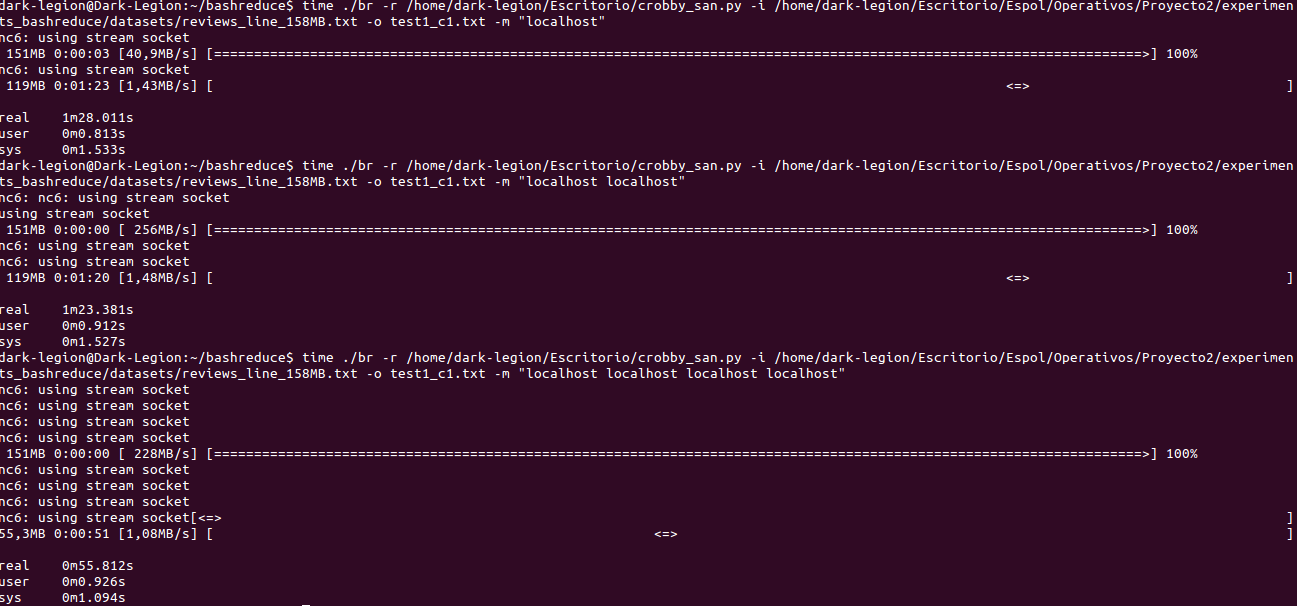
***Experimento Roberto Yoncón***



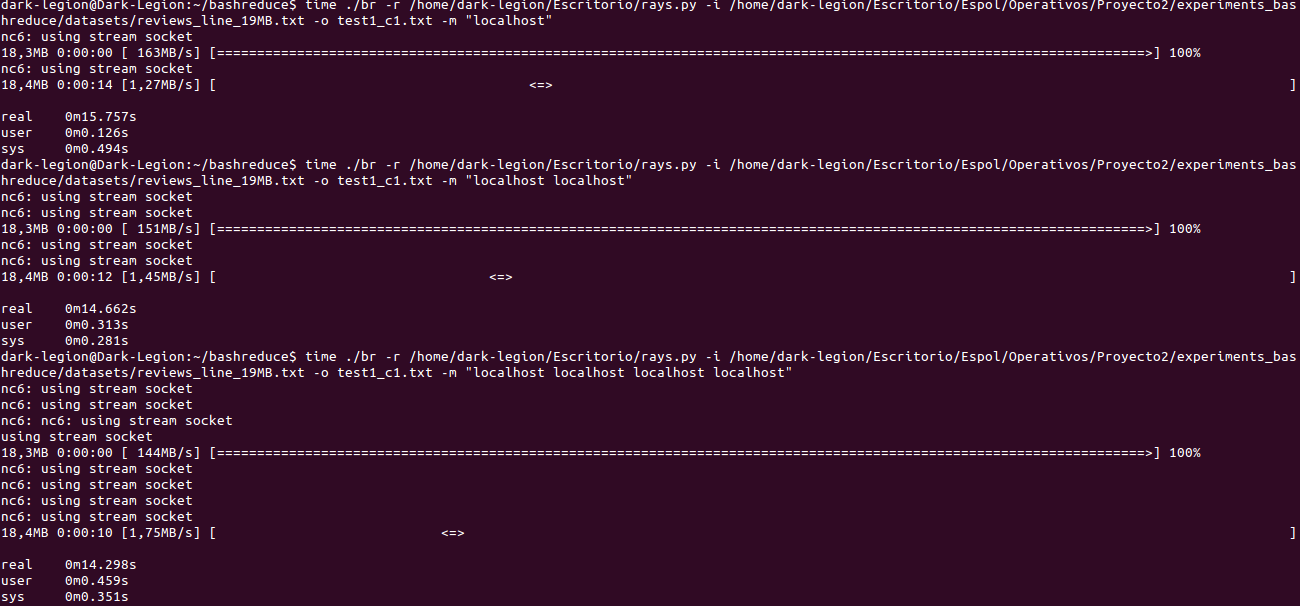


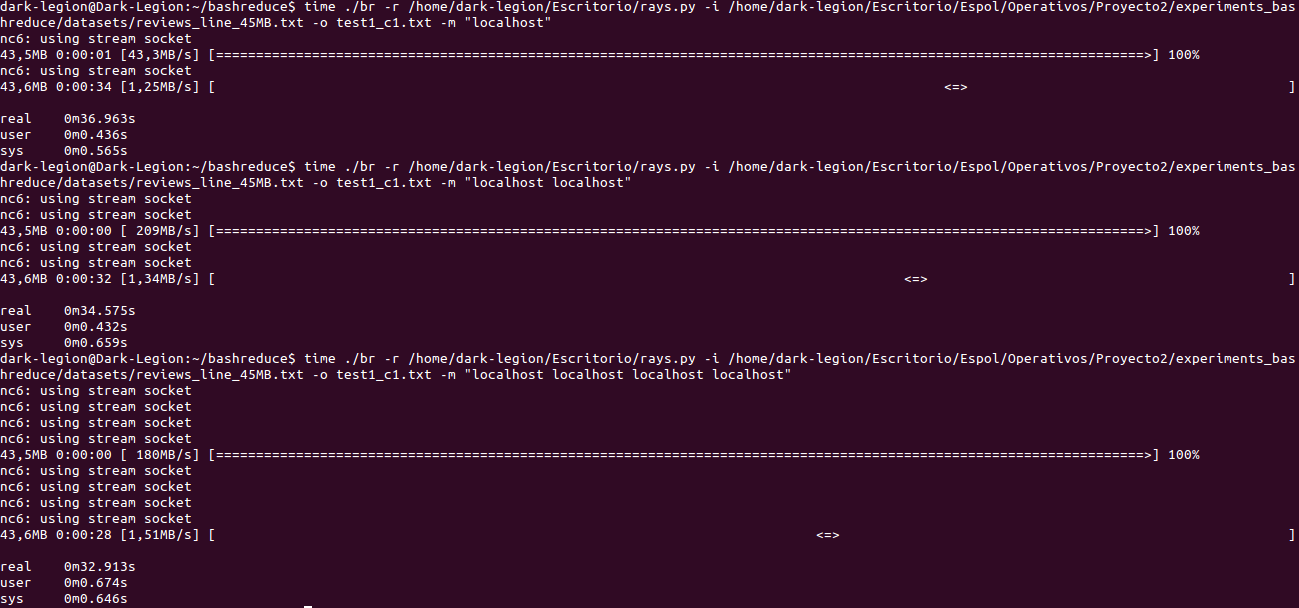


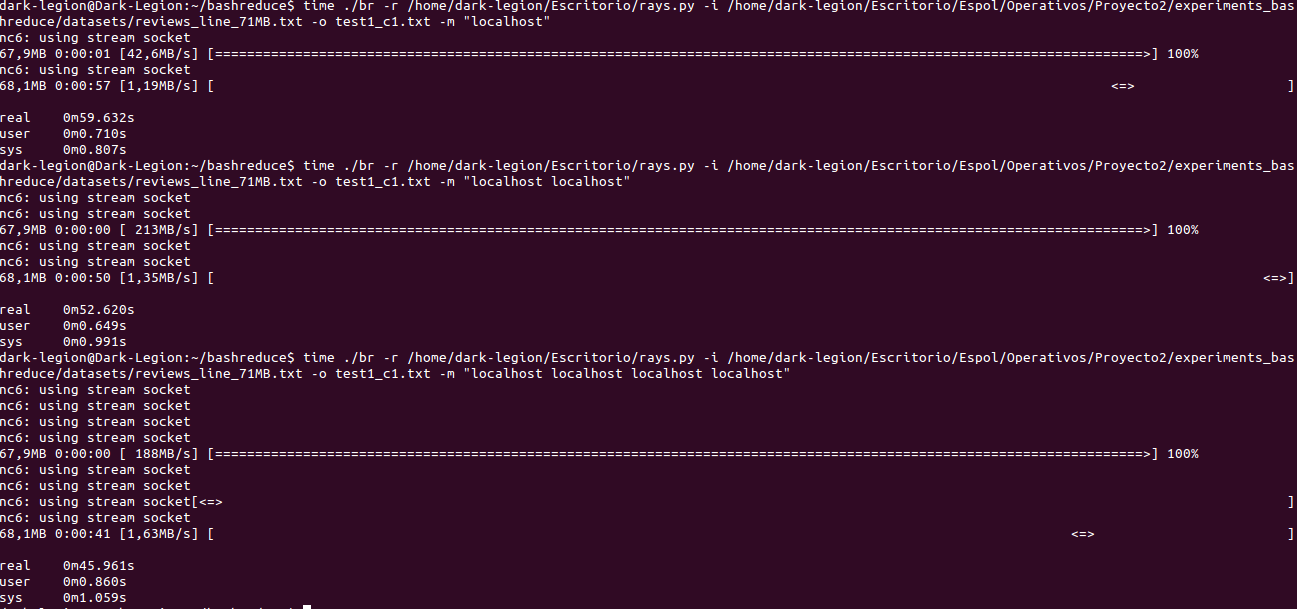


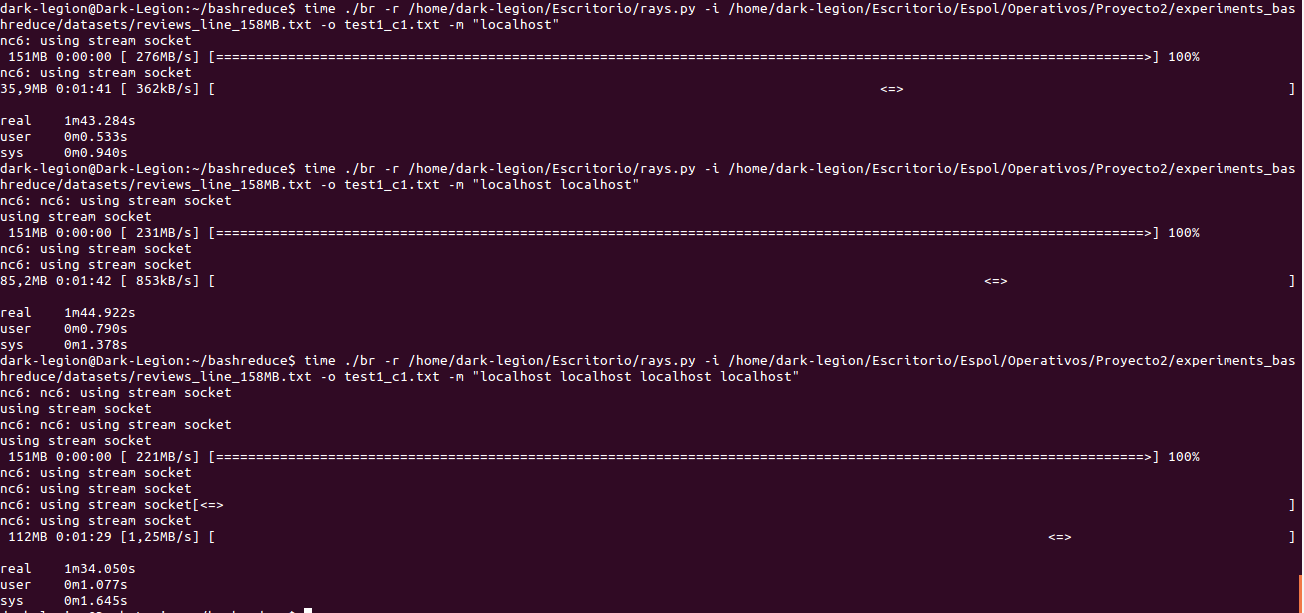


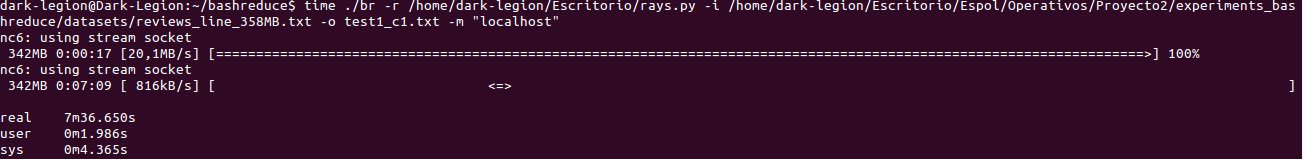
***Experimento Ray Montiel***

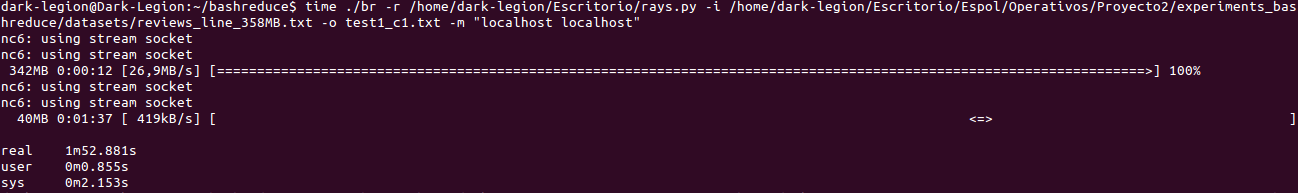


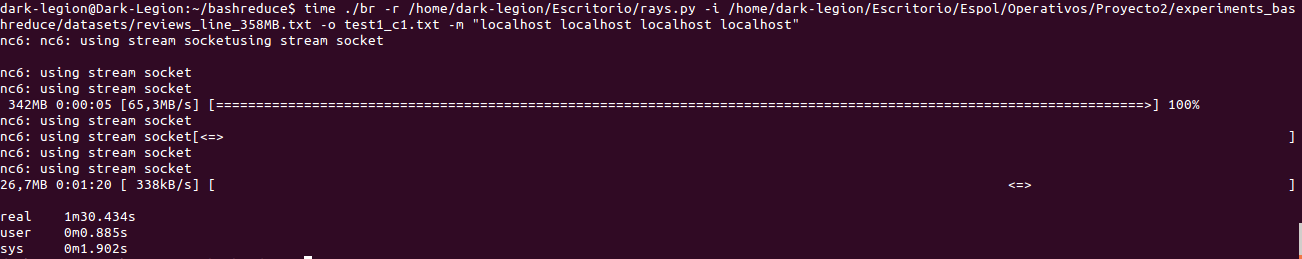












***Experimento Pedro Iñiguez***

