Universidad Nacional de San Agustin
Esc. Profesional de Ciencia de la Computación

Cloud Computing

Informe Contador de Palabras

Profesor: Alvaro Henry Mamani Aliaga
Alumno: Diego André Ranilla Gallegos
José Miguel Huamán Cruz

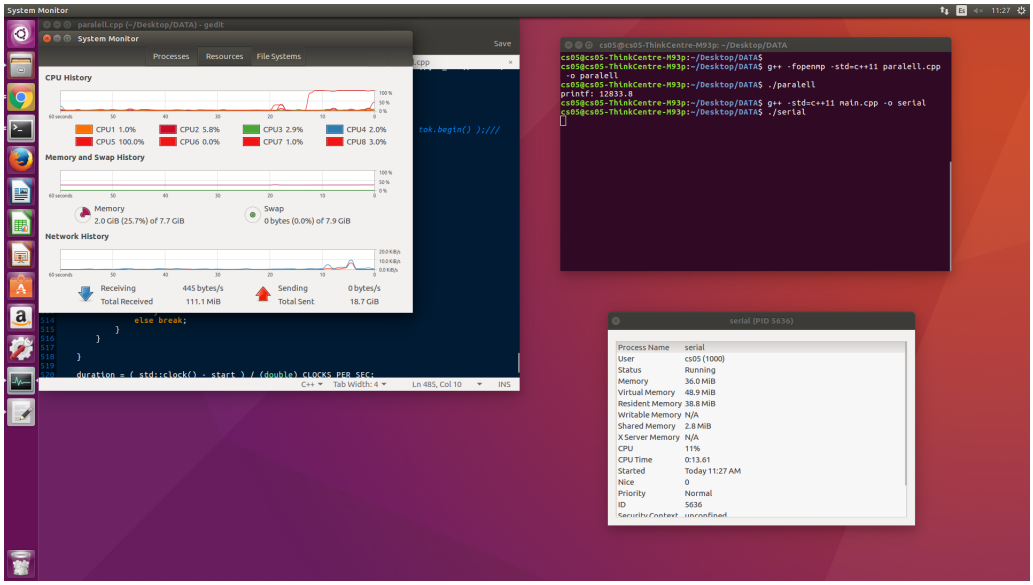
8 de Septiembre del 2017

1. Pruebas realizadas

Estas pruebas fueron hechas en un computador del laboratorio de Ciencia de la Computación la cual tiene las siguientes características:

- Memoria RAM 7,7 GiB
- Intel Core $i7-4770$ CPU, 3,40 GHz x 8
- Disco 453,9 GB

Para la prueba usamos una base de datos de entrada el cual pesa 50 GB sobre el cual se preprocesa, en este caso convertimos las letras mayúsculas en minúsculas mediante comandos Bash en la terminal (*$tr 'A-Z' 'a-z' < entrada.txt > destino.txt*).
El programa fue hecho en C++ en el cual se destaca el uso de la tabla Hash (*Unordered map* contenedor STL), con la finalidad para que pueda tanto registrar como contar las apariciones de una palabra en la base de datos de manera muy veloz, exactamente en un tiempo de ejecución $\theta(1)$, como salida nos retornará el tiempo de ejecución en segundos además un archivo (*salida.txt*) en el cual están las palabras contadas.



El resultado obtenido (representado en segundos) es el siguiente:

```
cs05@cs05-ThinkCentre-M93p:~/Desktop/DATA$ g++ -std=c++11 main.cpp -o serial
cs05@cs05-ThinkCentre-M93p:~/Desktop/DATA$ ./serial
printf: 1665.86
```

El tiempo de ejecución del contador de palabras 1665.86 segundos en otras palabras el programa se tarda en contar las palabras en 27.76 minutos.

1.1. Paralelización

Este código se puede paralelizar, para esta ocasión se emplea *OpenMP*. Con la finalidad que cada hilo tome cierta cantidad de la data total para su procesamiento, vamos preprocesar una vez más la data, separándola en diferentes archivos mediante los comandos Bash de la terminal.

Entonces en este código lo que podemos destacar son las directivas *pragma* que se van a emplear para la paralelización, para este programa empleamos dos directivas *pragma*:

- ♯*pragma omp parallel num_threads(n_files)*, este se emplea para asignarle a cada hilo un archivo que contiene una parte de la data total.
- ♯*pragma omp critical*, esta directiva se va emplear para que los hilos no se colisionen al acceder sobre la tabla Hash.