Procesamiento HPC utilizando OpenMP en Sistema SafeRoom

Agustin Balart, Cristian Famiglietti,Ernesto Alessandrini,Juan Cruz Rey, Rodrigo D'Amico.

Universidad Nacional de La Matanza,Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina
balartagustin@gmail.com,cristian.famy@gmail.com,eralesan48@gmail.com,
rodrigo.damico92@gmail.com

Resumen: La finalidad de este trabajo es describir la implementación de un procesamiento eficiente de un gran volúmen de información que llega al sistema **SafeRoom** utilizando HPC (paralelismo aplicado con OpenMP), en un ámbito con altos valores de monóxido de carbono (CO) y una necesidad de control riguroso de temperaturas y de CO.

Palabras claves: HCP , OpenMP, CO .

1 Introducción

- El objetivo de esta investigación está aplicada al sistema SafeRoom, un recinto con la tecnología de seguridad adecuada para operar bajo condiciones industriales y experimentales donde se encuentran niveles peligrosos de monóxido de carbono y se necesita controlar sin margen de error la temperatura.

En consecuencia de respirar altos niveles concentrados de CO por la combustión deficiente de sustancias como plásticos, y sin ventilación adecuada, se producen muchas muertes. Por ello es necesario medir con alta precisión dentro del recinto los valores de CO presentes en el aire durante las actividades que se desarrollen en el lugar.

En consecuencia de la manipulación de gases o líquidos que reaccionan de manera violenta con temperatura (como gases catalogados de peligro H240), es preciso tener un conocimiento de alta precisión sobre la temperatura que hay dentro del recinto durante las actividades del mismo, para evitar accidentes o fallos en los experimentos debido a que no se dispone con la información adecuada de la temperatura.

Teniendo estos dos motivos principales como motor de investigación, se procede a generar una adecuada y precisa medición de los valores CO y temperatura dentro del recinto, por medio de un considerable volumen de información que envían gran cantidad de sensores distribuidos uniformemente en el lugar. Esta información debe ser procesada por un sistema embebido, el cual para ejecutarse de manera más rápida y eficiente va a utilizar paralelismo a nivel código, implementando OpenMP.

2 Desarrollo

Antes de entrar en detalle en el algoritmo se detalla el funcionamiento de SafeRoom. En forma simplificada, el sistema embebido (multi core) recibe el muestreo de los sensores de monóxido de carbono y de los sensores de temperatura cada cierto intervalo de tiempo, dichos datos viajan por comunicación Bluetooth. Una vez que se reciben los datos continuamente se calcula el promedio de temperatura y el promedio del nivel de monóxido de carbono. Con estos datos se activarán o no los actuadores correspondientes y se alertará de la situación (alarma sonora,alarma lumínica,ventilador,servo motor para abrir la ventana).

Debido a que se disponen de 1000 sensores de temperatura y 1000 sensores de monóxido de carbono en la habitación, la intención es hacer uso de las directivas OpenMp con el fin de explotar el

[1] https://github.com/Eralesan/SafeRoom

paralelismo de cómputo de nuestro Arduino multicore y poder obtener un cálculo eficiente y preciso necesario por el ambiente crítico que se tiene.

2 Explicación del algoritmo.

El algoritmo a utilizar se basa en el modelo **fork-join**, que proviene de los sistemas Unix, donde una tarea muy pesada se divide en K hilos (fork) con menor peso, para luego "recolectar" sus resultados al final y unirlos en un solo resultado (join). En este caso, se dividirá el procesamiento en un hilo por cada 250 sensores de temperatura, luego se aplica el mismo concepto para los 1.000 sensores de monóxido, ya que tenemos 4 hilos al tener 4 cores en nuestro sistema. De esta forma el volúmen de datos es procesado paralelamente mediante una suma con procesamientos veloces en cada core. Luego de esa suma, se juntan los resultados en el programa principal, para hacer el correspondiente promedio y actuar en base a los valores finales obtenidos .

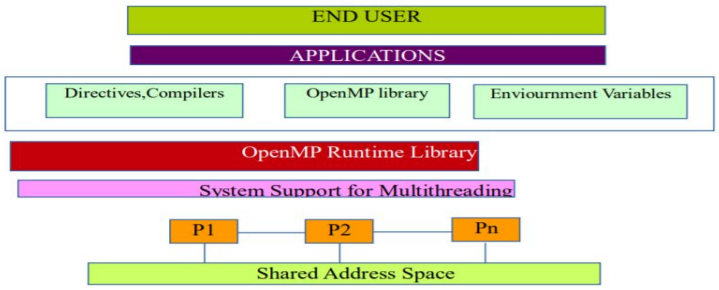

Figure 2.2 Basic Structure of OpenMP

Ejemplo en código C/C++:

```
#include <omp.h>
#include <stdio.h>
void main(){
        double temperaturas[1000];
        double monoxidos[1000];
/* previamente se toman los valores de todos los sensores y se los guardan en los vectores temperaturas[] y monoxidos[], este procedimiento se hace aplicando OpenMP para reducir el tiempo de ejecución utilizando paralelismo. Pero centrémonos en este ejemplo a continuación con el cálculo del promedio de los valores obtenidos.*/

        double sumaDeTemperatura,temperaturaPromedio = 0;
        double sumaDeMonoxido,monoxidoPromedio= 0;
        long long i;

        #pragma omp parallel num_threads(4)
    {
```

```
                #pragma omp for reduction(+:sumaDeTemperatura)
                for(i = 0;i < temperaturas.length;i++)
                sumaDeTemperatura+= temperaturas[i];
        }
        #pragma omp parallel num_threads(4)
      {
                #pragma omp for reduction(+:sumaDeMonoxido)
                for(i = 0;i < monoxidos.length;i++)
                sumaDeMonoxido+= monoxidos[i];
        }
/*El equipo de 4 hilos que se encuentra con los for ejecuta una o más fracciones
de iteraciones como resultado de dividir el bucle delimitado por la directiva
entre los 4 hilos*/
/* Lo que hace el modificador reduction es que asigna a cada hilo de ejecución una
copia de sumaDeTemperatura en un for y sumaDeMonoxido en el otro, de tal forma que
cada hilo modificará su propia versión de la variable. Cuando los 4 hilos
finalizan se suman las 4 versiones de la variable y el resultado se almacena en la
variable que hemos definido*/
/*Luego se guardan las variables en otras, para continuar con el programa*/

temperaturaPromedio=sumaDeTemperatura;
monoxidoPromedio=sumaDeMonoxido;

return;
}
```

Herramientas para usar OpenMP en Andriod:

Paso 1: Usar NDK para compilar C / C ++ para Android

1.1 Seleccione y descargue un paquete NDK para su plataforma de desarrollo de la siguiente página[3].

Paso 2: configurar banderas de compilador OpenMP

El NDK de Android utiliza el compilador cruzado gcc[4] para compilar los códigos fuente C / C ++ en binarios ejecutables de Android. Para habilitar el soporte de OpenMP en la compilación, edite el archivo de script Android.mk del proyecto Android NDK C / C ++ y agregue el modificador fopenmp[5] a la configuración del compilador y el enlazador:

```
LOCAL_CFLAGS + = -fopenmp
LOCAL_LDFLAGS + = -fopenmp
```

Referencias:

[3] Página de descarga de NDK

[4] La Colección de compiladores de GNU es un sistema compilador producido por el Proyecto GNU que admite varios lenguajes de programación.

[5] flag -fopenmp para compilar usando GCC

3 Pruebas que pueden realizarse

-Para conocer la verdadera diferencia en aplicar paralelismo o no hacerlo, tomar el tiempo que lleva ejecutar el programa en forma secuencial, sin uso de OpenMP. Luego ejecutarlo con OpenMP y comparar resultados.

-La medición de procesamiento del sistema en un ambiente con valores casi imperceptibles de monóxido de carbono para analizar la precisión de los sensores que es un requisito esencial en este sistema.

-Analizar la carga que puede soportar en extremo forzando a activar todos los sensores y que el sistema reaccione como se espera.

4 Conclusiones

En base a nuestro proyecto de investigación, SafeRoom necesitó de gran cantidad de sensores, para que informen de manera precisa el estado del ambiente en el que se encuentra. Una vez ingresada en el sistema esa información, se procedió a procesar con OpenMP en paralelo la gran cantidad de datos para obtener mejora en el tiempo de ejecución y que se pueda reducir el tiempo en que se toman los valores de los sensores.

Como pudimos observar en la investigación, trabajar HPC implementando OpenMP es una buena alternativa para optimizar los tiempos de ejecución para poder aprovechar al máximo el poder de procesamiento del hardware y la optimización del código cuando hay gran cantidad de información a procesar, evitando el uso de threads configurados manualmente y utilizándolos implícitamente gracias a las directivas del compilador, bibliotecas y variables de entorno que nos provee OpenMP, explotando el paralelismo de una manera muy efectiva.

Para futuros trabajos donde se pueda observar que el paralelismo es una buena opción, no hay que dudar en utilizar esta tecnología.

5 Referencias

1. Ashwini M. Bhugul, International Journal of Computer Science and Mobile Computing, Vol.6 Issue.2, February- 2017.
https://www.ijcsmc.com/docs/papers/February2017/V6I2201713.pdf

2. J. Ciesko, S. Mateo, X. Teruel, X. Martorell, E. Ayguade, J. Labarta, A. Duran, B. R. de Supinski, S. L. Olivier, K. Li, A. E. Eichenberger,Towards Task-Parallel Reductions in OpenMP,September 30, 2015 through October 2, 2015
https://e-reports-ext.llnl.gov/pdf/801361.pdf

3. Barcelona Supercomputing Center: OmpSs Specification (April, 25th 2014),
http: //pm.bsc.es/ompss-docs/specs