Procesamiento HPC en Sistema de automatizacion de luces

Molina Mariano, Sanchez Julian

[1]Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina
Marianoo.molina@gmail.com; jlysanchez@gmail.com

Resumen. El objetivo de esta investigación es el de realizar el procesamiento de los datos entregados por múltiples embebidos, dispersos en una casa, siendo necesario el análisis de los datos que estos entregan de manera simultánea. Para poder realizar esto se propone la utilización de OpenMP para hacer beneficio del paralelismo.

Palabras claves: OpenMP, HPC, Servidor.

1 Introducción

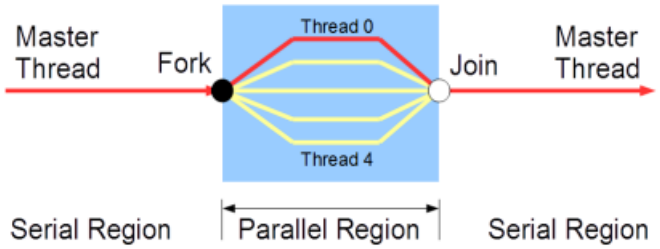
La investigación que se desarrolla en este documento esta aplicada al sistema LumusSystem, un sistema de iluminación inteligente que integra a todo un espacio. La problemática que se presenta es la del gran volumen de sensores que contiene una casa con múltiples habitaciones, por lo que un servidor, encargado del control general, podría saturarse o colapsar, haciendo que los tiempos de respuesta no sean los estipulados, como también podría realizar un incorrecto control sobre la iluminación de los distintos cuartos administrados.

Lo propuesto es hacer uso del paralelismo para el procesamiento de la información otorgada por cada espacio (lectura de los sensores), así lograr un correcto control por parte del servidor y mejorar los tiempos de respuesta de este. Esto se puede lograr implementando la plataforma de paralelismo OpenMP, el cual nos otorga portabilidad, simplicidad a la hora de implementar, y seguridad entre hilos. Otra implementación posible hubiese sido POSIX Threads, lo cual daría resultado, pero debido a su control de hilos de bajo nivel, sería necesario un riguroso desarrollo sobre la sincronización de hilos y la protección de memoria entre ellos. [1], [2]

2 Desarrollo

Nuestra investigación tiene como origen el despliegue de nuestro sistema LumusSystem, a un dominio más amplio. Actualmente el sistema realiza el control de la luminosidad de una sola habitación, teniendo como entrada las señales emitidas por sensores de luz, movimiento, sonido, en conjunto con un sistema de alarma programable. Debido a que el fin de esta investigación seria la implementación del control de múltiples cuartos, siendo así el análisis de cada habitación lo realizara un servidor común a todos los sensores y espacios; esto generaría un constante flujo de datos hacia el servidor, lo cual, ya observado anteriormente, afectaría los tiempos de respuesta de este como también fallos en el control del nivel de luz de cada cuarto.

Para la implementación de dicha tecnología es necesario contar con un sistema multiprocesador para la ejecución en paralelo de los distintos espacios. Un proceso inicial comenzaría la ejecución de la región serie, conocido como "máster", hasta este momento no existe paralelismo. Luego en base a las operaciones "fork/join" que nos otorga la programación de OpenMp [3], [4] se comienza la distribución del recibo de información de manera paralela, donde cada hilo tendrá asignado una habitación de la cual tomará los datos y realizará el análisis correspondiente a cada sensor.



3 Explicación del algoritmo.

Los valores de lectura de cada embebido son respecto al nivel de luz que recibe el ambiente desde el exterior, como también si se registra el ingreso de una persona, además de contar con una alarma programable y un detector de sonido para activar y desactivar las luces. En base a los distintos valores que puede recibir de un embebido el sistema realiza múltiples acciones, como accionar la persiana, encender o apagar las luces, o hacer sonar la alarma. Si en algún momento de la ejecución ocurre algún error el sistema lo notificara.

```
void main(){
    //Suponiendo que en este caso se cuenta con cuatro habitaciones automatizadas
    int th_id, nthreads=4;
    //Valores que envian los distintos embebidos en cada habitacion
    int[4] valoresLectura;
    int accion;
    string embebido;
    string error = null;
    //inicio de parte paralelizada
    #pragma omp parallel private(th_id , valoresLectura, embebido, error, accion)
    while(error){
        th_id = omp_get_thread_num();
        //Dependiendo de que hilo se esta ejecutando obtengo los valores del embebido
        // que corresponda
        switch(th_id){
            case 0:
                embebido = "direccion del embebido uno";
                break;
            case 1:
                embebido = "direccion del embebido dos";
                break;
            case 2:
                embebido = "direccion del embebido tres";
                break;
            case 3:
                embebido = "direccion del embebido cuatro";
                break;
        }

        obtenerValores(embebido, &valoresLectura, &error);
        analizarValores(&valoresLectura, &accion, &error);
        realizarAccion(&accion, &error);
        if(error != null)
            informarError(&error);
    }
}
```

```
void obtenerValores( embebido, valoresLectura,  error){
    //obtengo los valores del embedico correspondiente a la direccion
    // definida segun el hilo
    //Cargo los valores en la variable valoresLectura
    //Si hay algun error lo informo en la variable
}
void analizarValores( valoresLectura, accion, error){
    //analizo los valores obtenidos del embebido
    //determino una accion a realizar (de ser necesario)
    //si hay algun error lo informo en la variable
}

void realizarAccion( accion, embebido, error){
    //realizo la accion correspondiente en base al valor de la accion
    //si hay algun error lo informo en la variable
}

void informarError( error){
    //informo el error encontrado
    send("Error %s ocurrido por el hilo %d",error,omp_get_thread_num())
}
```

4 Pruebas que pueden realizarse

Como prueba principal a realizar, sería la medición de tiempos de respuesta que se obtienen del servidor aplicando el análisis de manera secuencial, es decir, de a una habitación a la vez, y luego implementar paralelismo con OpenMp. En base a los tiempos obtenidos realizar una comparación. En estas pruebas se asume que el dominio es el mismo.

Otras pruebas pueden ser como influye en el rendimiento del servidor, a medida que se incrementan la cantidad de habitaciones de las cuales recibe información. Así poder realizar un análisis sobre los tiempos de respuesta que se obtiene utilizando OpenMP para el paralelismo, para cada caso implementado.

Por último, seria aplicar una sobre carga de datos desde múltiples puntos, generando un caso de fatiga, para así determinar los límites del sistema y si este es capaz de funcionar con normalidad en aquellos casos.

5 Conclusiones

A través de este trabajo intentamos aumentar el alcance con el que cuenta nuestro sistema de iluminación inteligente LumusSystem, incrementando la cantidad de habitaciones que el sistema es capaz de administrar (actualmente una sola), utilizando para ello OpenMP, lo que nos permite implementar de forma sencilla y segura el paralelismo para el procesamiento de los distintos ambientes.

Durante la investigación pudimos aprender cómo resolver situaciones en las que un mismo código que será ejecutado de manera paralela, sea capaz de recibir información y comunicarse con distintos embebidos.

Para futuros trabajos se podría implementar con el uso de regiones críticas, el registro de los distintos valores que se fueron registrando en cada habitación, como puede ser la cantidad de personas que ingresaron a los distintos espacios, en un determinado tiempo o periodo.

6 Referencias

1.http://www.dalkescientific.com/writings/diary/archive/2012/01/13/openmp_vs_posix_threads.html ”OpenMp vs POSIX threads”

2. http://www.cs.colostate.edu/~cs675/OpenMPvsThreads.pdf "OpenMp vs threads”

3. https://www.ijcsmc.com/docs/papers/February2017/V6I2201713.pdf “Parallel Computing using OpenMP”

4. https://www.openmp.org/wp-content/uploads/omp-hands-on-SC08.pdf "A Hands-on Introduction to OpenMP"