Procesamiento HPC en Sistema de control de habitación integrado

Beckerman, Leandro Martín; López, Pablo Joel; De Diego, Brian Adrián; Fritz, Braian Nicolás

Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina
leobeckerman88@gmail.com; pablo.lopez3895@gmail.com; briandediego@gmail.com; braian.fritz@gmail.com

Resumen. El objetivo de esta investigación es de realizar el procesamiento de un gran volumen de datos generado por la implementación del sistema en diversos complejos de universidades por lo que será necesario poder manejar de forma simultánea el acceso a las habitaciones. Para poder realizar esto se propone la utlización de OpenMP para la explotación del paralelismo.

Palabras claves: HPC, BigData, Multicore, OpenMp, Server

1 Introducción

La investigación que se desarrolla en este documento está aplicada al Sistema de Control de Habitación Integrado. La problemática que se presenta es la del control de acceso de una gran cantidad de individuos de forma simultánea, por lo que un servidor podría saturarse o colapsar si este tuviera que manejar todas estas peticiones de forma secuencial, lo que podría provocar que, por ejemplo, el usuario que intente acceder primero a la habitación, se le otorgue el acceso último debido a las condiciones de carrera provocadas por el acceso simultáneo de una gran cantidad de usuarios.

Lo que se propone es la utilización de paralelismo para el procesamiento eficiente de dichas peticiones, mejorando el tiempo de respuesta del servidor mediante el modelo de programación OpenMP, ya que se utiliza un único servidor (memoria compartida).

2 Desarrollo

En principio, es necesario explicar el funcionamiento de nuestro sistema de control de acceso. Su funcionamiento consiste en la utilización de un sistema embebido que recibe una lectura a través de un módulo RFID y luego de i interpretarla los datos viajan por comunicación Wi-Fi desde el embebido al servidor central de proceso. Una vez allí se evaluará si dicho acceso está permitido o no, brindando una respuesta acorde al embebido el cual permitirá o no el acceso del usuario a la habitación. Debido a que una institución académica posee una cantidad aproximada de 650 habitaciones entre los cuales podemos nombrar aulas, laboratorios y centros de control; se pueden presentar una gran cantidad de peticiones simultáneas al servidor, como se mencionó anteriormente lo que provocara demoras o hasta incluso fallos en la apertura de las puertas.

Para el desarrollo de esta tecnología es imperativo contar con un sistema multiprocesador para la paralelización de los distintos procesos que van a controlar el acceso de los usuarios. Para ello se utilizarán distintos algoritmos que encolarán a las peticiones de los usuarios y las distribuirán entre distintos procesos a través de las directivas de fork/join proporcionadas por el modelo de programación OpenMP. También es necesario contar con una base de datos que pueda trabajar de forma paralela.

3 Explicación del algoritmo.

```
Void main() {
   Event,on("nuevo usuario", function(user) {
         pushUsuarioEnCola(user);
         responderAccesoUsuarioConcurrente (user);
   });
}

Function responderAccesoUsuarioConcurrente (user) {
   // Comienzo de parte de codigo OpenMP
   if(isServerSaturado() && fork()) {
         respuestaUsuario(getUsuariosDeCola(users))
   // FIN de parte de codigo OpenMP
   } else {

         If(tieneAcceso(users[user]))  {
               Send("valid user")
         } else  {
               Send("invalid user")
         }
   }
```

```
}

Function respuestaUsuario(users) {
        For(int i = 0; i < users.length; i++) {
                If(tieneAcceso(users[user]))  {
                        Send(“valid user”)
                } else  {
                        Send(“invalid user”)
                }
        }
}
```

4 Pruebas que pueden realizarse

Una serie de pruebas que se pueden realizar es la prueba de tiempo de respuesta, con solamente una habitación y luego realizar la misma prueba con 650 habitaciones y evaluar la diferencia utilizando OpenMP en ambas pruebas.

Realizar la prueba con OpenMP y sin OpenMP en un entorno real para conocer la verdadera diferencia entre aplicar paralelismo o no.

Realizar un caso de fatiga que involucre a todos los embebidos a la vez, realicen las solicitudes de acceso constantemente durante un lapso de 20 segundos y que el sistema responda de forma esperada.

5 Conclusiones

En base a la investigación en el proyecto pudimos observar que en un entorno académico existen una gran cantidad de habitaciones o aulas en las que se puede aplicar el sistema de control de acceso.

Cuando hay un gran volumen de datos a procesar, OpenMP mejora los tiempos de ejecución del servidor, proporcionando una mejora en el tiempo de respuesta que recibe el usuario al intentar acceder una habitación utilizando el sistema de control de acceso, siendo esto uno de los principales requerimientos para una excelente experiencia de usuario.

Sin embargo, por mas que se paralelicen las peticiones y con ellas las consultas a la base de datos, aun queda mucho por mejorar, debido a que los tiempos de respuesta de un acceso a almacenamiento tradicional siempre sera el eslabon mas lento de la cadena. Esto sin embargo podria ser optimizado utilizando discos de estado solido en vez de discos duros mecanicos, obeniendo notorias mejoras en el tiempo de respuesta.

Por otro lado se podria levantar toda la informacion necesaria para el sistema en memoria, y paralelizar las comprobaciones logicas, evitando asi el acceso a un medio de almacenamiento. Sin embargo es algo que queda fuera de este docuemnto.

Por lo que concluimos que OpenMP es una buena opción para usarlo en nuestro proyecto como lo hemos descripto o para sus posibles mejoras y aplicarlo a entornos más complejos como edificios urbanos, centros médicos y instituciones académicas.

6 Referencias

1. Modeling and Implementation of an Asynchronous Approach to Integrating HPC and Big Data Analysis (2016) https://ac.els-cdn.com/S1877050916306482/1-s2.0-S1877050916306482-main.pdf?_tid=39ae1f83-ceee-41d6-9eff-0524a6353525&acdnat=1531155385_36c2200487276b56d7ed9236ee28d923

2. Measuring synchronization and cheduling overheads in openMp (Enero 2016) https://www.epcc.ed.ac.uk/sites/default/files/PDF/ewomp99paper.pdf

3. Paralel computing using openMp (Febrero 2017) https://www.ijcsmc.com/docs/papers/February2017/V6I2201713.pdf