Laboratorio de Algoritmos Paralelos

Luis Gustavo Cáceres Zegarra

April 4, 2017

1 Introducción

La capacidad de cálculo de los microprocesadores ha crecido en los ultimos años, aunque el incremento en la velocidad del procesador no ha ido a l apar con el incremento en la velocidad de memoria. En problemas reales muchas veces los datasetes utilizados son muy grandes, para lo cual necesitamos utilizar los recursos de la computadora de la mejor manera posible con la finalidad de mejorar el tiempo de respuesta utilizando algoritmos eficaces.

En objetivo de este trabajo es ver como al utilizar dos metodos de multiplicacion de matrices, uno de ellos deberia mostrar mejor performance. Los dos metodos o técnicas que utilizaremos son:

- Método Secuencial
- Optimización por bloques

2 Método Secuencial

En el cual se utiliza tres for anidados, esta técnica es muy rudimentaria y su tiempo de ejecución es muy costoso. Tambien para matrices muy grandes podria ocasionarnos problemas.

```
void mult__secuencial_matrix(int m1[][m_size],int m2[][m_size],int r[][m_size],int n)
{
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            r[i][j]=0;
            for(int k=0;k<n;k++)
            {
                r[i][j]+=m1[i][k]*m2[k][j];
            }
        }
    }
}
```

3 Optimización por bloques

Esta técnica esta basada en el mismo método secuencial, solo que en lugar de utilizar elementos unitarios utilizaremos bloques de la matriz.

```
void block_mult(int m1[][m_size],int m2[][m_size],int r[][m_size],int n,int block_num)
{
    int block_size = n/block_num;
```

N	Secuencial	Bloques
200	74445	70968
400	603130	561290

Table 1: Resultados pruebas.

```
    for(int ii=0;ii<n;ii+=block_size)
    {
        for(int jj=0;jj<n;jj+=block_size)
        {
            for(int kk=0;kk<n;kk+=block_size)
            {
                for(int i=ii;i<ii+block_size;i++)
                {
                    for(int j=jj;j<jj+block_size;j++)
                    {
                        for(int k=kk;k<kk+block_size;k++)
                        {
                            r[i][j] += m1[i][k]*m2[k][j];
                        }
                    }
                }
            }
        }
    }

}
```

4 Pruebas

En la tabla 1 se mostrara los resultados que se obtuvieron al ejecutar los dos algoritmos con matrices cuadradas de 200 y 400. Para la medición de tiempos utilizamos la libreria Chrono de C. Como su puede ver hubo una pequeña mejora cuando se utilizan matrices de grandes dimensiones y el algoritmo por bloques.

5 Conclusiones

El algoritmo de multiplicación de matrices por bloques probo un mejor performance que el algortimo lineal. Para matrices pequeñas es casi inperceptible la diferencia pero con matrices de mayor tamaño es mas notorio. Otra ventaja es que al utilizar multiplicación por bloques nos evitamos que el tamaño de la matriz sea mas grande que la cache,