INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

"Tarea 2. Uso eficiente de la memoria cache"

Alumno:
Lara Cázares Jaime Arturo

Materia:
Desarrollo de sistemas distribuidos

Grupo:
4CM3

Profesor: Pineda Guerrero Carlos

Índice general

1. Código fuente.

1.1. MultiplicaMatriz

En este primer programa se multiplican directamente la matriz A y B por lo que hay que mover muchas más veces los renglones (filas) de la matriz B y dado a que las matrices se manejan por renglones en Java se debe cargar muchas veces los datos a la Cache por lo que hace ineficiente la ejecución. En la línea 23 se puede observar lo antes mencionado.

A continuación se muestran los códigos utilizados, la diferencia entre ellos solo es el valor de N (tamaño de la matriz cuadrada) a utilizar.

1.1.1. N=100

```
1  class MultiplicaMatriz{
2      static int N = 100;//Tamanio de matrices
3      static int[][] A = new int[N][N];//Declaracion dematriz de NxN
4      static int[][] B = new int[N][N];//Declaracion dematriz de NxN
5      static int[][] C = new int[N][N];//Declaracion dematriz de NxN
6
7      public static void main(String[] args){
8          long t1 = System.currentTimeMillis();
9          // inicializa las matrices A y B
10         System.out.println("N="+N);
11         for (int i = 0; i < N; i++){
12             for (int j = 0; j < N; j++){
13                 A[i][j] = 2 * i - j;
14                 B[i][j] = i + 2 * j;
15                 C[i][j] = 0;
16             }
17         }
18         // multiplica la matriz A y la matriz B, el resultado queda en la matriz C
19
20         for (int i = 0; i < N; i++){
21             for (int j = 0; j < N; j++){
22                 for (int k = 0; k < N; k++){
23                     C[i][j] += A[i][k] * B[k][j];//Uso ineficiente de Cache
24                 }
25             }
26         }
27         long t2 = System.currentTimeMillis();
28         System.out.println("Tiempo: " + (t2 - t1) + "ms");
29     }
30 }
```

1.1.2. N=200

```
1  class MultiplicaMatriz{
2      static int N = 200;//Tamanio de matrices
3      static int[][] A = new int[N][N];//Declaracion dematriz de NxN
4      static int[][] B = new int[N][N];//Declaracion dematriz de NxN
5      static int[][] C = new int[N][N];//Declaracion dematriz de NxN
6
7      public static void main(String[] args){
8          long t1 = System.currentTimeMillis();
9          // inicializa las matrices A y B
10         System.out.println("N="+N);
11         for (int i = 0; i < N; i++){
12             for (int j = 0; j < N; j++){
13                 A[i][j] = 2 * i - j;
14                 B[i][j] = i + 2 * j;
```

```
                C[i][j] = 0;
            }
        }
        // multiplica la matriz A y la matriz B, el resultado queda en la matriz C

        for (int i = 0; i < N; i++){
            for (int j = 0; j < N; j++){
                for (int k = 0; k < N; k++){
                    C[i][j] += A[i][k] * B[k][j];//Uso ineficiente de Cache
                }
            }
        }
        long t2 = System.currentTimeMillis();
        System.out.println("Tiempo: " + (t2 - t1) + "ms");
    }
}
```

1.1.3. N=300

```
class MultiplicaMatriz{
    static int N = 300;//Tamanio de matrices
    static int[][] A = new int[N][N];//Declaracion dematriz de NxN
    static int[][] B = new int[N][N];//Declaracion dematriz de NxN
    static int[][] C = new int[N][N];//Declaracion dematriz de NxN

    public static void main(String[] args){
        long t1 = System.currentTimeMillis();
        // inicializa las matrices A y B
        System.out.println("N="+N);
        for (int i = 0; i < N; i++){
            for (int j = 0; j < N; j++){
                A[i][j] = 2 * i - j;
                B[i][j] = i + 2 * j;
                C[i][j] = 0;
            }
        }
        // multiplica la matriz A y la matriz B, el resultado queda en la matriz C

        for (int i = 0; i < N; i++){
            for (int j = 0; j < N; j++){
                for (int k = 0; k < N; k++){
                    C[i][j] += A[i][k] * B[k][j];//Uso ineficiente de Cache
                }
            }
        }
        long t2 = System.currentTimeMillis();
        System.out.println("Tiempo: " + (t2 - t1) + "ms");
    }
}
```

1.1.4. N=500

```
class MultiplicaMatriz{
    static int N = 500;//Tamanio de matrices
    static int[][] A = new int[N][N];//Declaracion dematriz de NxN
    static int[][] B = new int[N][N];//Declaracion dematriz de NxN
    static int[][] C = new int[N][N];//Declaracion dematriz de NxN

    public static void main(String[] args){
        long t1 = System.currentTimeMillis();
        // inicializa las matrices A y B
        System.out.println("N="+N);
        for (int i = 0; i < N; i++){
            for (int j = 0; j < N; j++){
                A[i][j] = 2 * i - j;
```

```
                B[i][j] = i + 2 * j;
                C[i][j] = 0;
            }
        }
        // multiplica la matriz A y la matriz B, el resultado queda en la matriz C

        for (int i = 0; i < N; i++){
            for (int j = 0; j < N; j++){
                for (int k = 0; k < N; k++){
                    C[i][j] += A[i][k] * B[k][j];//Uso ineficiente de Cache
                }
            }
        }
        long t2 = System.currentTimeMillis();
        System.out.println("Tiempo: " + (t2 - t1) + "ms");
    }
}
```

1.1.5. N=1000

```
class MultiplicaMatriz{
    static int N = 1000;//Tamanio de matrices
    static int[][] A = new int[N][N];//Declaracion dematriz de NxN
    static int[][] B = new int[N][N];//Declaracion dematriz de NxN
    static int[][] C = new int[N][N];//Declaracion dematriz de NxN

    public static void main(String[] args){
        long t1 = System.currentTimeMillis();
        // inicializa las matrices A y B
        System.out.println("N="+N);
        for (int i = 0; i < N; i++){
            for (int j = 0; j < N; j++){
                A[i][j] = 2 * i - j;
                B[i][j] = i + 2 * j;
                C[i][j] = 0;
            }
        }
        // multiplica la matriz A y la matriz B, el resultado queda en la matriz C

        for (int i = 0; i < N; i++){
            for (int j = 0; j < N; j++){
                for (int k = 0; k < N; k++){
                    C[i][j] += A[i][k] * B[k][j];//Uso ineficiente de Cache
                }
            }
        }
        long t2 = System.currentTimeMillis();
        System.out.println("Tiempo: " + (t2 - t1) + "ms");
    }
}
```

1.2. MultiplicaMatriz2

El objetivo de este código es la mejor utilización de la memoria cache y lograr un menor tiempo de ejecución que el programa anterior, para ello se pretende cargar los renglones y recorrerlos en la Cache para que se maneje una **localidad espacial**.

Para seguir con la lógica de la multiplicación de matrices es necesario calcular la **transpuesta** de la *Matriz B*, entonces de la línea 20 a 26 se hace esta operación que consiste básicamente en "voltear" la matriz para que se accedan a sus datos de forma en la que los índices se puedan intercambiar y mejorar el rendimiento de la ejecución.

Por lo tanto, el recorrido se queda prácticamente igual solo que ahora se cargan los renglones de la **Matriz B** en la cache y se obtenga una mejor eficiencia de la memoria Cache. Este cambio se observa en la línea 33 del código.

A continuación se muestran los códigos utilizados, la diferencia entre ellos solo es el valor de N (tamaño de la matriz cuadrada) a utilizar.

1.2.1. N=100

```
1  class MultiplicaMatriz_2{
2      static int N = 100;//Tamanio de matrices
3      static int[][] A = new int[N][N];//Declaracion dematriz de NxN
4      static int[][] B = new int[N][N];//Declaracion dematriz de NxN
5      static int[][] C = new int[N][N];//Declaracion dematriz de NxN
6
7      public static void main(String[] args){
8          long t1 = System.currentTimeMillis();
9          // inicializa las matrices A y B
10         System.out.println("N="+N);
11         for (int i = 0; i < N; i++){
12             for (int j = 0; j < N; j++){
13                 A[i][j] = 2 * i - j;
14                 B[i][j] = i + 2 * j;
15                 C[i][j] = 0;
16             }
17         }
18         // transpone la matriz B, la matriz traspuesta queda en B
19
20         for (int i = 0; i < N; i++){
21             for (int j = 0; j < i; j++){
22                 int x = B[i][j];
23                 B[i][j] = B[j][i];
24                 B[j][i] = x;
25             }
26         }
27         // multiplica la matriz A y la matriz B, el resultado queda en la matriz C
28         // notar que los indices de la matriz B se han intercambiado
29
30         for (int i = 0; i < N; i++){
31             for (int j = 0; j < N; j++){
32                 for (int k = 0; k < N; k++){
33                     C[i][j] += A[i][k] * B[j][k];//Uso eficiente de Cache
34                 }
35             }
36         }
37
38         long t2 = System.currentTimeMillis();
39         System.out.println("Tiempo: " + (t2 - t1) + "ms");
40     }
41 }
```

1.2.2. N=200

```
class MultiplicaMatriz_2{
    static int N = 200;//Tamanio de matrices
    static int[][] A = new int[N][N];//Declaracion dematriz de NxN
    static int[][] B = new int[N][N];//Declaracion dematriz de NxN
    static int[][] C = new int[N][N];//Declaracion dematriz de NxN

    public static void main(String[] args){
        long t1 = System.currentTimeMillis();
        // inicializa las matrices A y B
        System.out.println("N="+N);
        for (int i = 0; i < N; i++){
            for (int j = 0; j < N; j++){
                A[i][j] = 2 * i - j;
                B[i][j] = i + 2 * j;
                C[i][j] = 0;
            }
        }
        // transpone la matriz B, la matriz traspuesta queda en B

        for (int i = 0; i < N; i++){
            for (int j = 0; j < i; j++){
                int x = B[i][j];
                B[i][j] = B[j][i];
                B[j][i] = x;
            }
        }
        // multiplica la matriz A y la matriz B, el resultado queda en la matriz C
        // notar que los indices de la matriz B se han intercambiado

        for (int i = 0; i < N; i++){
            for (int j = 0; j < N; j++){
                for (int k = 0; k < N; k++){
                    C[i][j] += A[i][k] * B[j][k];//Uso eficiente de Cache
                }
            }
        }

        long t2 = System.currentTimeMillis();
        System.out.println("Tiempo: " + (t2 - t1) + "ms");
    }
}
```

1.2.3. N=300

```
class MultiplicaMatriz_2{
    static int N = 300;//Tamanio de matrices
    static int[][] A = new int[N][N];//Declaracion dematriz de NxN
    static int[][] B = new int[N][N];//Declaracion dematriz de NxN
    static int[][] C = new int[N][N];//Declaracion dematriz de NxN

    public static void main(String[] args){
        long t1 = System.currentTimeMillis();
        // inicializa las matrices A y B
        System.out.println("N="+N);
        for (int i = 0; i < N; i++){
            for (int j = 0; j < N; j++){
                A[i][j] = 2 * i - j;
                B[i][j] = i + 2 * j;
                C[i][j] = 0;
            }
        }
        // transpone la matriz B, la matriz traspuesta queda en B

        for (int i = 0; i < N; i++){
            for (int j = 0; j < i; j++){
```

```
                int x = B[i][j];
                B[i][j] = B[j][i];
                B[j][i] = x;
            }
        }
        // multiplica la matriz A y la matriz B, el resultado queda en la matriz C
        // notar que los indices de la matriz B se han intercambiado

        for (int i = 0; i < N; i++){
            for (int j = 0; j < N; j++){
                for (int k = 0; k < N; k++){
                    C[i][j] += A[i][k] * B[j][k];//Uso eficiente de Cache
                }
            }
        }

        long t2 = System.currentTimeMillis();
        System.out.println("Tiempo: " + (t2 - t1) + "ms");
    }
}
```

1.2.4. N=500

```
class MultiplicaMatriz_2{
    static int N = 500;//Tamanio de matrices
    static int[][] A = new int[N][N];//Declaracion dematriz de NxN
    static int[][] B = new int[N][N];//Declaracion dematriz de NxN
    static int[][] C = new int[N][N];//Declaracion dematriz de NxN

    public static void main(String[] args){
        long t1 = System.currentTimeMillis();
        // inicializa las matrices A y B
        System.out.println("N="+N);
        for (int i = 0; i < N; i++){
            for (int j = 0; j < N; j++){
                A[i][j] = 2 * i - j;
                B[i][j] = i + 2 * j;
                C[i][j] = 0;
            }
        }
        // transpone la matriz B, la matriz traspuesta queda en B

        for (int i = 0; i < N; i++){
            for (int j = 0; j < i; j++){
                int x = B[i][j];
                B[i][j] = B[j][i];
                B[j][i] = x;
            }
        }
        // multiplica la matriz A y la matriz B, el resultado queda en la matriz C
        // notar que los indices de la matriz B se han intercambiado

        for (int i = 0; i < N; i++){
            for (int j = 0; j < N; j++){
                for (int k = 0; k < N; k++){
                    C[i][j] += A[i][k] * B[j][k];//Uso eficiente de Cache
                }
            }
        }

        long t2 = System.currentTimeMillis();
        System.out.println("Tiempo: " + (t2 - t1) + "ms");
    }
}
```

1.2.5. N=1000

```
class MultiplicaMatriz_2{
    static int N = 1000;//Tamanio de matrices
    static int[][] A = new int[N][N];//Declaracion dematriz de NxN
    static int[][] B = new int[N][N];//Declaracion dematriz de NxN
    static int[][] C = new int[N][N];//Declaracion dematriz de NxN

    public static void main(String[] args){
        long t1 = System.currentTimeMillis();
        // inicializa las matrices A y B
        System.out.println("N="+N);
        for (int i = 0; i < N; i++){
            for (int j = 0; j < N; j++){
                A[i][j] = 2 * i - j;
                B[i][j] = i + 2 * j;
                C[i][j] = 0;
            }
        }
        // transpone la matriz B, la matriz traspuesta queda en B

        for (int i = 0; i < N; i++){
            for (int j = 0; j < i; j++){
                int x = B[i][j];
                B[i][j] = B[j][i];
                B[j][i] = x;
            }
        }
        // multiplica la matriz A y la matriz B, el resultado queda en la matriz C
        // notar que los indices de la matriz B se han intercambiado

        for (int i = 0; i < N; i++){
            for (int j = 0; j < N; j++){
                for (int k = 0; k < N; k++){
                    C[i][j] += A[i][k] * B[j][k];//Uso eficiente de Cache
                }
            }
        }

        long t2 = System.currentTimeMillis();
        System.out.println("Tiempo: " + (t2 - t1) + "ms");
    }
}
```

2. Gráfica de dispersión Multiplicarmatriz vs Multiplicarmatriz2

En la Figura 1 se puede observar la gráfica del tiempo de ejecución de ambos programas, donde el programa MultiplicaMatriz2 (en color naranja) es más eficiente que el programa Multiplicamatriz (color azul), esto es más notable mientras que N se vuelve más grande.

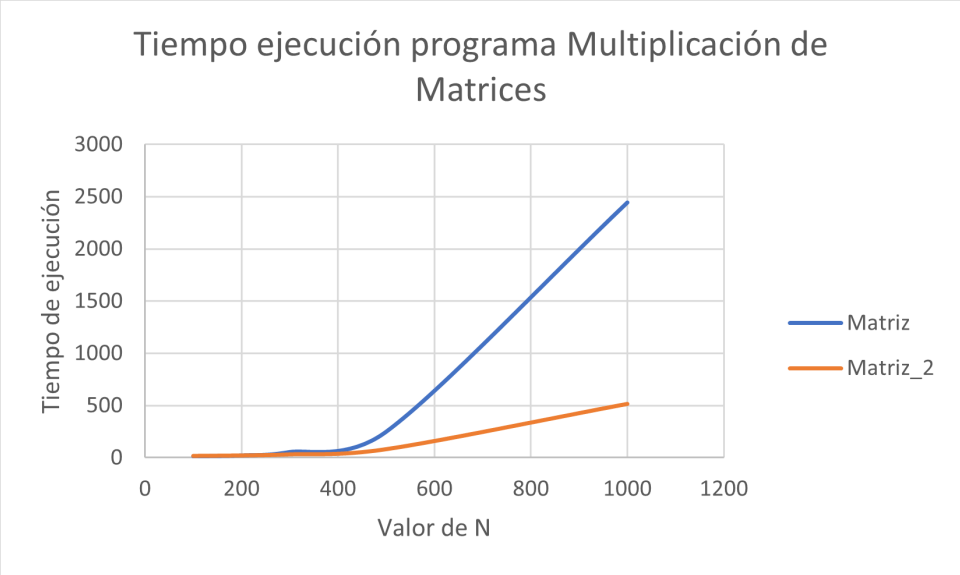

Figura 1: Gráfica de dispersión de Multiplicarmatriz(color azul) vs Multiplicarmatriz2(color naranja).

3. Datos del hardware utilizado.

- Marca de procesador: Intel.
- Modelo de procesador: i7 7500U.
- RAM: 16 GB.
- Cache: L2=512 kb y L3=4096kb