



## MINISTÉRIO DA EDUCAÇÃO

Universidade Federal de Itajubá - Campus Itabira  
Rua Irmã Ivone Drummond, 200 - Distrito Industrial II  
Itabira - MG - 35903-087

**MATRÍCULA:** 28614 \_\_\_\_\_ **NOME:** Douglas Venâncio \_\_\_\_\_

**MATRÍCULA:** 31455 \_\_\_\_\_ **NOME:** Yan Zingra Pereira \_\_\_\_\_

### Paralelismo de Dados e Controle

Neste trabalho foram feitos paralelismos de controle e dados de dois algoritmos, o primeiro algoritmo se trata de uma soma de senos, cossenos e exponenciais gerados a partir da série de Taylor e o segundo algoritmo se trata da multiplicação de duas matrizes. No primeiro algoritmo, para realizar o paralelismo por dados, foi feita a divisão do número de somatória pelo número de threads e cada thread fazia a soma de uma parcela das somas, que posteriormente foram somadas para se obter o resultado final. Já no paralelismo por controle o que foi dividido foram as operações seno, cosseno e exponencial, somando cada parte do somatório em cada uma das threads e depois foi feita a soma do resultado de cada thread.

Para realizar o paralelismo de dados no segundo algoritmo foi realizado a divisão do número de linhas da primeira matriz pelo número de threads, e cada uma foi responsável por uma parcela das linhas. O paralelismo por controle não foi feito devido na operação de multiplicação de matrizes não conseguirmos dividir a tarefa de controle, sem fazer com que o código fique sequencial, ou seja, uma thread ficar esperando a outra acabar, para prosseguir. Para utilizar controle nesta situação seria necessário a utilização de uma abordagem mista com paralelismo de dados com paralelismo de controle utilizando pipeline onde quando uma thread termina-se sua multiplicação outra viria somando enquanto as outras threads permanecem fazendo as multiplicações, mas esta abordagem resultaria em um desempenho pior que o paralelismo de dados, devido a tarefa de multiplicação demanda relativamente mais processamento que a de soma logo é mais vantajoso designar essa thread para fazer o paralelismo de dados.

### Resultados

#### Algoritmo 1-Soma de Seno, Cosseno e Exponenciais:

Como foi possível observar nas figuras abaixo, quanto maior o número de threads menor o tempo de execução, porém após o número de threads que o processador utilizado possui não há mais melhora no desempenho podendo ocorrer uma pequena queda de desempenho quando comparado com a utilização exata do número de threads que o processador possui, devido a necessidade do processador organizar quais threads irão executar primeiro, por ter o número de núcleos inferior ao número de threads.

Abaixo são mostrados os tempos de execução:



## MINISTÉRIO DA EDUCAÇÃO

Universidade Federal de Itajubá - Campus Itabira  
Rua Irmã Ivone Drummond, 200 - Distrito Industrial II  
Itabira - MG - 35903-087

```
douglinha5k@douglinha5k-To-be-filled-by-O-E-M: ~/Desktop/ParalelosTrab1/Somatorio
File Edit View Search Terminal Help
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~$ cd Desktop/
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop$ cd ParalelosTrab1/Somatorio/
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somatorio/
o$ java Somatorio
Error: Could not find or load main class Somatorio
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somatorio/
o$ java Sequencial
N = 2000          Tempo = 6237 ms
N = 4000          Tempo = 46815 ms
N = 6000          Tempo = 160294 ms
N = 8000          Tempo = 396609 ms
N = 10000         Tempo = 792783 ms
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somatorio/
o$
```

Figura 1: Algoritmo 1 Sequencial



## MINISTÉRIO DA EDUCAÇÃO

Universidade Federal de Itajubá - Campus Itabira  
Rua Irmã Ivone Drummond, 200 - Distrito Industrial II  
Itabira - MG - 35903-087

```
douglinha5k@douglinha5k-To-be-filled-by-O-E-M: ~/Desktop/ParalelosTrab1/Somatorio
File Edit View Search Terminal Help
N = 2000      Resultado = 8091.586890 Tempo = 3675 ms
^Cdouglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somator
$ javac ParaleloDados.java
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somatorio
$ java ParaleloDados
Entre com a quantidade de threads:
3
N = 2000      Tempo = 3659 ms
N = 4000      Tempo = 30346 ms
N = 6000      Tempo = 104198 ms
N = 8000      Tempo = 248298 ms
^Cdouglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somator
$ javac ParaleloDados.java
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somatorio
$ java ParaleloDados
Entre com a quantidade de threads:
2
N = 2000      Resultado = 8091.586890 Tempo = 4394 ms
N = 4000      Resultado = 8091.588890 Tempo = 34884 ms
N = 6000      Resultado = 8091.590890 Tempo = 125158 ms
N = 8000      Resultado = 8091.592890 Tempo = 300225 ms
N = 10000     Resultado = 8091.594890 Tempo = 592786 ms
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somatorio
$
```

Figura 2: Algoritmo 1 Paralelismo de Dados [2 Threads]

```
$ java ParaleloDados
Entre com a quantidade de threads:
4
N = 2000      Resultado = 8091.586890 Tempo = 3212 ms
^Cdouglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somator
$ java ParaleloDados
Entre com a quantidade de threads:
4
N = 2000      Resultado = 8091.586890 Tempo = 3195 ms
N = 4000      Resultado = 8091.588890 Tempo = 25885 ms
N = 6000      Resultado = 8091.590890 Tempo = 90740 ms
N = 8000      Resultado = 8091.592890 Tempo = 215260 ms
N = 10000     Resultado = 8091.594890 Tempo = 413512 ms
```

Figura 3: Algoritmo 1 Paralelismo de Dados [4 Threads]



## MINISTÉRIO DA EDUCAÇÃO

Universidade Federal de Itajubá - Campus Itabira  
Rua Irmã Ivone Drummond, 200 - Distrito Industrial II  
Itabira - MG - 35903-087

```
douglinha5k@douglinha5k-To-be-filled-by-O-E-M: ~/Desktop/ParalelosTrab1/Somatorio
File Edit View Search Terminal Help
8
N = 2000      Resultado = 8091.586890 Tempo = 2610 ms
^Cdouglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somator
$ java ParaleloDados
Entre com a quantidade de threads:
8
N = 2000      Resultado = 8091.586890 Tempo = 2656 ms
^Cdouglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somator
$ java ParaleloDados
Entre com a quantidade de threads:
12
N = 2000      Resultado = 8091.586890 Tempo = 2482 ms
N = 4000      Resultado = 8091.588890 Tempo = 18384 ms
^C^Cdouglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somat
$ java ParaleloDados
Entre com a quantidade de threads:
8
N = 2000      Resultado = 8091.586890 Tempo = 2686 ms
N = 4000      Resultado = 8091.588890 Tempo = 19776 ms
N = 6000      Resultado = 8091.590890 Tempo = 65987 ms
N = 8000      Resultado = 8091.592890 Tempo = 167155 ms
N = 10000     Resultado = 8091.594890 Tempo = 331992 ms
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somatorio
$
```

Figura 3: Algoritmo 1 Paralelismo de Dados [8 Threads]



## MINISTÉRIO DA EDUCAÇÃO

Universidade Federal de Itajubá - Campus Itabira  
Rua Irmã Ivone Drummond, 200 - Distrito Industrial II  
Itabira - MG - 35903-087

```
douglinha5k@douglinha5k-To-be-filled-by-O-E-M: ~/Desktop/ParalelosTrab1/Somatorio
File Edit View Search Terminal Help
$ javac ParaleloControle.java
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somatorio
$ javac ParaleloDados.java
error: Class names, 'ParaleloControle', are only accepted if annotation processing is explicitly requested
1 error
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somatorio
$ java ParaleloControle
N = 2000          Tempo = 2774 ms
N = 4000          Tempo = 21608 ms
^Cdouglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somatorio
$ java ParaleloDados
Entre com a quantidade de threads:
1
N = 2000          Resultado = 8091.586890 Tempo = 5473 ms
^Cdouglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somatorio
$ java ParaleloControle
N = 2000          Tempo = 2766 ms
N = 4000          Tempo = 21567 ms
N = 6000          Tempo = 76281 ms
N = 8000          Tempo = 184270 ms
N = 10000         Tempo = 361168 ms
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Somatorio
$
```

Figura 4: Algoritmo 1 Paralelismo de Controle

Abaixo são mostrados os códigos do algoritmo 1:

### Função Fatorial:

```
import java.math.BigDecimal;

/*
 * Funcoes matematicas utilitarias
 */
public class Util {

    /*
     * Funcao para calcular o fatorial de um numero qualquer n
     */
    public static BigDecimal Fatorial(BigDecimal n) {
        BigDecimal soma = BigDecimal.ONE;

        //calcula de 0!
        if (n.compareTo(BigDecimal.ZERO) == 0)
        {
            return soma;
        }
    }
}
```



## MINISTÉRIO DA EDUCAÇÃO

Universidade Federal de Itajubá - Campus Itabira  
Rua Irmã Ivone Drummond, 200 - Distrito Industrial II  
Itabira - MG - 35903-087

```
        else{
            //calculo de n!
            for(BigDecimal i = BigDecimal.ONE; i.compareTo(n.add(BigDecimal.ONE)) != 0; i =
i.add(BigDecimal.ONE))
            {
                soma = soma.multiply(i);
            }
            return soma;
        }
    }

    /*
     * Funcao para calcular a potencia de 2^n
     */
    public static BigDecimal Potencia2(BigDecimal n) {
        BigDecimal soma = BigDecimal.ONE;

        //calculo de 2^0
        if (n.compareTo(BigDecimal.ZERO) == 0) return soma;

        //calculo de 2^n
        for(BigDecimal i = BigDecimal.ONE; i.compareTo(n) != 0; i =
i.add(BigDecimal.ONE))
            soma = soma.multiply(BigDecimal.valueOf(2));

        return soma;
    }
}
```

### Algoritmo 1 :

```
import java.math.BigDecimal;
import java.math.RoundingMode;

public class Sequencial {

    public BigDecimal Somatorio(BigDecimal n, BigDecimal x) {
        BigDecimal soma = BigDecimal.ZERO;
        BigDecimal menos1 = new BigDecimal("-1");
        BigDecimal dois = new BigDecimal("2");

        for(BigDecimal i = BigDecimal.ZERO; i.compareTo(n) != 0; i =
i.add(BigDecimal.ONE))
        {
            soma = soma.add(
                Util.Fatorial(i), 6, RoundingMode.UP));
            soma = soma.add(
                (((menos1).pow(i.intValue())).multiply(x.pow((i.intValue()*2)))
                .divide(
                    Util.Fatorial(i), 6, RoundingMode.UP));
            );
        }
    }
}
```



## MINISTÉRIO DA EDUCAÇÃO

Universidade Federal de Itajubá - Campus Itabira  
Rua Irmã Ivone Drummond, 200 - Distrito Industrial II  
Itabira - MG - 35903-087

```
.divide(Util.Fatorial(i.multiply(dois)),6,RoundingMode.UP)    );
    soma = soma.add(
(((menos1).pow(i.intValue()).multiply(x.pow(((i.intValue()*2)+1)))
).divide(Util.Fatorial((i.multiply(dois)).add(BigDecimal.ONE)),6,RoundingMode.UP)
));
    }

    return soma;
}

public static void main(String[] args) {
    BigDecimal n, resultado, passo;
    BigDecimal x=new BigDecimal("9");
    long tempo1, tempo2;
    Sequential alg = new Sequential();

    n = BigDecimal.ZERO;
    passo = new BigDecimal("2000");

    //Calcula o somatorio para os valores 2000, 4000, 6000, 8000, 10000
    for(int i = 0; i < 5; i++){
        n = n.add(passo)
        ;
        tempo1 = System.nanoTime();
        resultado = alg.Somatorio( n,x );
        tempo2 = System.nanoTime();

        System.out.print("N = " + n.toString());
        System.out.println("\tTempo = " + String.valueOf((tempo2 -
tempo1)/1000000) + " ms");
    }

}

}
```

### Algoritmo 1 [controle] :

```
import java.math.BigDecimal;
import java.math.RoundingMode;

public class ParaleloControle extends Thread {
    private BigDecimal n; //numero de elementos para processamento
    private BigDecimal x;
    private BigDecimal menos1;
    private BigDecimal dois;
    private BigDecimal resultado; //resultado do processamento
    private int tarefa; //tarefa realizada pela thread

    /*
    * Construtor padrao
    */
}
```





## MINISTÉRIO DA EDUCAÇÃO

Universidade Federal de Itajubá - Campus Itabira  
Rua Irmã Ivone Drummond, 200 - Distrito Industrial II  
Itabira - MG - 35903-087

```
*/
public ParaleloControle(BigDecimal n, BigDecimal x, int tarefa){
    this.n = n;
    this.tarefa = tarefa;
    this.x=x;
    menos1=new BigDecimal("-1");
    dois=new BigDecimal("2");
    resultado = BigDecimal.ZERO;
}

/*
 * Algoritmo executado pela thread
 * @see java.lang.Thread#run()
 */
public void run() {
    resultado = BigDecimal.ZERO;

    if (tarefa == 0) { //exponencial

        for(BigDecimal i = BigDecimal.ZERO; i.compareTo(n) != 0; i =
i.add(BigDecimal.ONE))
        {
            resultado = resultado.add(
                (x.pow(i.intValue()))
            ).divide(
                Util.Fatorial(i),6,RoundingMode.UP);
        }

    }
    else if(tarefa == 1) { //cos

        for(BigDecimal i = BigDecimal.ZERO; i.compareTo(n) != 0; i =
i.add(BigDecimal.ONE))
            resultado = resultado.add(
                (((menos1).pow(i.intValue())).multiply(x.pow((i.intValue()*2)))
            ).divide(Util.Fatorial(i.multiply(dois)),6,RoundingMode.UP)
            );

        }
    else //sen
    {
        for(BigDecimal i = BigDecimal.ZERO; i.compareTo(n) != 0; i =
i.add(BigDecimal.ONE))
        {
            resultado = resultado.add(
                (((menos1).pow(i.intValue())).multiply(x.pow(((i.intValue()*2)+1)))
            ).divide(Util.Fatorial((i.multiply(dois)).add(BigDecimal.ONE)),6,RoundingMode.UP)
            );
        }
    }

}

/*
 * Retorna o resultado do processamento
 */
public BigDecimal Resultado() {
    return resultado;
}

public static void main(String[] args) throws InterruptedException {
```





## MINISTÉRIO DA EDUCAÇÃO

Universidade Federal de Itajubá - Campus Itabira  
Rua Irmã Ivone Drummond, 200 - Distrito Industrial II  
Itabira - MG - 35903-087

```
BigDecimal n,x, resultado, passo;
long tempo1, tempo2;

n = BigDecimal.ZERO;
passo = new BigDecimal("2000");
//X
x=new BigDecimal("9");

ParaleloControle thread0, thread1,thread2;

//Calcula o somatorio para os valores 2000, 4000, 6000, 8000, 10000
for(int i = 0; i < 5; i++){
    n = n.add(passo);
    resultado = BigDecimal.ZERO;

    //cria as threads
    thread0 = new ParaleloControle(n , x,0);
    thread1 = new ParaleloControle(n , x,1);
    thread2 = new ParaleloControle(n , x,2);

    tempo1 = System.nanoTime();

    //inicia as threads
    thread0.start();
    thread1.start();
    thread2.start();

    //define um ponto de sincronismo (barreira)
    //aguarda o termino do processamento de todas as threads
    thread0.join();
    thread1.join();
    thread2.join();

    //junta os resultados das threads
    resultado = resultado.add(thread0.Resultado());
    resultado = resultado.add(thread1.Resultado());
    resultado = resultado.add(thread2.Resultado());

    tempo2 = System.nanoTime();

    System.out.print("N = " + n.toString());
    System.out.println("\tTempo = " + String.valueOf((tempo2 -
tempo1)/1000000) + " ms");
}
}
}
```

### Algoritmo 1 [dados] :

```
import java.util.Scanner;
import java.math.BigDecimal;
```



## MINISTÉRIO DA EDUCAÇÃO

Universidade Federal de Itajubá - Campus Itabira  
Rua Irmã Ivone Drummond, 200 - Distrito Industrial II  
Itabira - MG - 35903-087

```
import java.math.RoundingMode;

public class ParaleloDados extends Thread {
    private BigDecimal inicio; //inicio do intervalo de processamento
    private BigDecimal fim; //fim do intervalo de processamento
    private BigDecimal resultado; //resultado do processamento
    private BigDecimal menos1=new BigDecimal("-1");
    private BigDecimal dois=new BigDecimal("2");
    private BigDecimal x=new BigDecimal("9");//x
    /*
     * Construtor padrao
     */
    public ParaleloDados(){
        this.inicio = BigDecimal.ZERO;
        this.fim = BigDecimal.ZERO;
        resultado = BigDecimal.ZERO;
    }

    /*
     * Define o inicio e o fim do intervalo de dados para processamento
     */
    public void DefineIntervalo(BigDecimal inicio, BigDecimal fim){
        this.inicio = inicio;
        this.fim = fim;
        resultado = BigDecimal.ZERO;
    }

    /*
     * Algoritmo executado pela thread
     * @see java.lang.Thread#run()
     */
    public void run() {
        resultado = BigDecimal.ZERO;

        for(BigDecimal i = inicio; i.compareTo(fim) != 0; i = i.add(BigDecimal.ONE)) {
            resultado = resultado.add(
                (x.pow(i.intValue()))
                .divide(
                    Util.Fatorial(i),6,RoundingMode.UP));
            resultado = resultado.add(
                (((menos1).pow(i.intValue())).multiply(x.pow((i.intValue()*2)))
                ).divide(Util.Fatorial(i.multiply(dois)),6,RoundingMode.UP)
                );
            resultado = resultado.add(
                (((menos1).pow(i.intValue())).multiply(x.pow(((i.intValue()*2)+1)))
                ).divide(Util.Fatorial((i.multiply(dois)).add(BigDecimal.ONE)),6,RoundingMode.UP)
                );
        }

    }

    /*
     * Retorna o resultado do processamento
     */
    public BigDecimal Resultado() {
        return resultado;
    }
}
```



## MINISTÉRIO DA EDUCAÇÃO

Universidade Federal de Itajubá - Campus Itabira  
Rua Irmã Ivone Drummond, 200 - Distrito Industrial II  
Itabira - MG - 35903-087

```
public static void main(String[] args) throws InterruptedException {
    Scanner teclado = new Scanner(System.in);
    int numThreads;
    BigDecimal n, resultado, passo, tamanho, posicao;
    long tempo1, tempo2;

    n = BigDecimal.ZERO;
    passo = new BigDecimal("2000");

    System.out.println("Entre com a quantidade de threads: ");
    numThreads = teclado.nextInt();

    teclado.close();

    ParaleloDados[] threads = new ParaleloDados[numThreads];

    //Calcula o somatorio para os valores 2000, 4000, 6000, 8000, 10000
    for(int i = 0; i < 5; i++){
        n = n.add(passo);
        tamanho = n.divide( BigDecimal.valueOf(numThreads) );

        posicao = BigDecimal.ONE;
        resultado = BigDecimal.ZERO;

        //cria as threads
        for(int j = 0; j < numThreads; j++){
            threads[j] = new ParaleloDados();

            tempo1 = System.nanoTime();

            //divide os dados entre as (numThreads - 1) threads
            for(int j = 0; j < numThreads - 1; j++) {
                threads[j].DefineIntervalo(posicao, posicao.add(tamanho));
                posicao = posicao.add(tamanho);
            }
            //atribui o restante dos dados para a ultima thread
            //resolve a questao da divisao inteira
            threads[numThreads - 1].DefineIntervalo(posicao, n);

            //inicia as threads
            for(int j = 0; j < numThreads; j++){
                threads[j].start();

                //define um ponto de sincronismo (barreira)
                //aguarda o termino do processamento de todas as threads
                for(int j = 0; j < numThreads; j++){
                    threads[j].join();

                    //junta os resultados das threads
                    for(int j = 0; j < numThreads; j++){
                        resultado = resultado.add(threads[j].Resultado());
                    }

                    tempo2 = System.nanoTime();

                    System.out.print("N = " + n.toString());
                    System.out.print("\tResultado = " + resultado);
                    System.out.println("\tTempo = " + String.valueOf((tempo2 -
```



## MINISTÉRIO DA EDUCAÇÃO

Universidade Federal de Itajubá - Campus Itabira  
Rua Irmã Ivone Drummond, 200 - Distrito Industrial II  
Itabira - MG - 35903-087

```
tempo1)/1000000) + " ms");  
    }  
}  
  
}
```

### Algoritmo 2-Multiplicação de Matrizes:

Neste algoritmo primeiramente foi feito a multiplicação de matriz de forma sequencial e após utilizando paralelismo de dados conforme foi aumentando o número de threads foi observando um grande aumento no desempenho até chegar no número de threads do processador não tendo mais ganho após isso.

Abaixo são mostrados os tempos de execução:

```
douglinha5k@douglinha5k-To-be-filled-by-O-E-M: ~/Desktop/ParalelosTrab1/Matriz  
File Edit View Search Terminal Help  
ava Sequencial  
^Cdouglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Matriz$  
ava Sequencial  
^[A^Cdouglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Mat  
avac Sequencial.java  
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Matriz$ j  
ava Sequencial  
^Cdouglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Matriz$  
ava Sequencial  
^[A^[A^Cdouglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1  
avac Sequencial.java  
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Matriz$ j  
ava Sequencial  
    Tempo = 2041 ms  
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Matriz$ j  
ava Sequencial  
    Tempo = 2040 ms  
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Matriz$ j  
ava Sequencial  
    Tempo = 2107 ms  
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Matriz$ j  
ava Sequencial  
    Tempo = 2179 ms  
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Matriz$
```

Figura 5: Algoritmo 2 Sequencial



## MINISTÉRIO DA EDUCAÇÃO

Universidade Federal de Itajubá - Campus Itabira  
Rua Irmã Ivone Drummond, 200 - Distrito Industrial II  
Itabira - MG - 35903-087

```
douglinha5k@douglinha5k-To-be-filled-by-O-E-M: ~/Desktop/ParalelosTrab1/Matriz
File Edit View Search Terminal Help
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Matriz$ j
ava ParaleloDados
Entre com a quantidade de threads:
1
    Tempo = 2037 ms
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Matriz$ j
ava ParaleloDados
Entre com a quantidade de threads:
2
    Tempo = 1027 ms
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Matriz$ j
ava ParaleloDados
Entre com a quantidade de threads:
3
    Tempo = 698 ms
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Matriz$ j
ava ParaleloDados
Entre com a quantidade de threads:
4
    Tempo = 579 ms
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Matriz$ j
ava ParaleloDados
Entre com a quantidade de threads:
5
    Tempo = 644 ms
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Matriz$ j
ava ParaleloDados
Entre com a quantidade de threads:
7
    Tempo = 567 ms
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Matriz$ j
ava ParaleloDados
Entre com a quantidade de threads:
8
    Tempo = 565 ms
douglinha5k@douglinha5k-To-be-filled-by-O-E-M:~/Desktop/ParalelosTrab1/Matriz$ j
```

Figura 6: Algoritmo 2 Paralelismo de Dados



## Algoritmo 2 [Sequencial]:

```
import java.math.BigDecimal;
import java.math.RoundingMode;

public class Sequencial {

    public void multiplica(int linha1,int coluna1,int linha2,int coluna2,int
matA[][],int matB[][],int matresp[][]){

        for (int i=0; i<linha1; i++)
        {
            for (int j=0 ; j<coluna2; j++)
            {
                for(int k=0 ; k<linha2 ; k++)
                matresp[i][j] += matA[i][k]*matB[k][j];
            }
        }
    }

    public static void main(String[] args) {
        long tempo1, tempo2;
        Sequencial mult = new Sequencial();
        int linha1=1000;
        int coluna1=1000;
        int linha2=coluna1;//LxN*NxC
        int coluna2=1000;
        int matA[][]=new int[linha1][coluna1];
        int matB[][]=new int[linha2][coluna2];
        int matresp[][]=new int[linha1][coluna2];

        //System.out.print("matA\n");
        for (int l = 0; l<linha1; l++) {
            for (int c = 0; c<coluna1; c++) {
                matA[l][c] = (int)(Math.random()*100);
                //System.out.print(matA[l][c]+ " ");
            }
            //System.out.println();
        }

        //System.out.println();
        //System.out.println();

        //System.out.print("matB\n");
        for (int l = 0; l<linha2; l++) {
            for (int c = 0; c<coluna2; c++) {
                matB[l][c] = (int)(Math.random()*100);
                //System.out.print(matB[l][c]+ " ");
            }
            //System.out.println();
        }
    }
}
```



## MINISTÉRIO DA EDUCAÇÃO

Universidade Federal de Itajubá - Campus Itabira  
Rua Irmã Ivone Drummond, 200 - Distrito Industrial II  
Itabira - MG - 35903-087

```
}  
//System.out.println();  
//System.out.println();  
  
tempo1 = System.nanoTime();  
  
mult.multiplica(linha1,coluna1,linha2,coluna2,matA,matB,matresp);  
  
tempo2 = System.nanoTime();  
  
//System.out.print("Matriz Resultante\n");  
for (int l = 0; l<linha1; l++) {  
    for (int c = 0; c<coluna2; c++) {  
        //System.out.print(matresp[l][c]+ " ");  
    }  
    //System.out.println();  
}  
  
System.out.println("\tTempo = " + String.valueOf((tempo2 - tempo1) /  
1000000) + " ms");  
  
}  
}
```

### Algoritmo 2 [dados] :

```
import java.util.Scanner;  
import java.math.BigDecimal;  
import java.math.RoundingMode;  
  
public class ParaleloDados extends Thread {
```





## MINISTÉRIO DA EDUCAÇÃO

Universidade Federal de Itajubá - Campus Itabira  
Rua Irmã Ivone Drummond, 200 - Distrito Industrial II  
Itabira - MG - 35903-087

```
private int iniciolinha;
private int fimlinha;
private int linha1;
private int coluna1;
private int linha2;
private int coluna2;
private int matA[][];
private int matB[][];
private int matresp[][];
/*
 * Construtor padrao
 */
public ParaleloDados(int linha1,int coluna1,int linha2,int coluna2,int matA[],[],int
matB[],[],int matresp[][]){
    this.linha1=linha1;
    this.coluna1=coluna1;
    this.linha2=linha2;
    this.coluna2=coluna2;
    this.matA=matA;
    this.matB=matB;
    this.matresp=matresp;
}

/*
 * Define o inicio e o fim do intervalo de dados para processamento
 */
public void DefineIntervalo(int iniciolinha, int fimlinha){
    this.iniciolinha = iniciolinha;
    this.fimlinha = fimlinha;
}

public void run() {

    for (int i=iniciolinha; i<fimlinha; i++)
    {
        for (int j=0 ; j<coluna2; j++)
        {
            for(int k=0 ; k<linha2 ; k++)
            matresp[i][j] += matA[i][k]*matB[k][j];
        }
    }

}

public static void main(String[] args) throws InterruptedException {
    Scanner teclado = new Scanner(System.in);
    long tempo1, tempo2;
    int linha1=1000;
    int coluna1=1000;
    int linha2=coluna1;//LxN*NxC
    int coluna2=1000;
    int matA[][]=new int[linha1][coluna1];
    int matB[][]=new int[linha2][coluna2];
    int matresp[][]=new int[linha1][coluna2];
```



## MINISTÉRIO DA EDUCAÇÃO

Universidade Federal de Itajubá - Campus Itabira  
Rua Irmã Ivone Drummond, 200 - Distrito Industrial II  
Itabira - MG - 35903-087

```
int numThreads;

//System.out.print("matA\n");
for (int l = 0; l<linha1; l++) {
    for (int c = 0; c<coluna1; c++) {
        matA[l][c] = (int)(Math.random()*100);
        //System.out.print(matA[l][c]+ " ");
    }
    //System.out.println();
}

//System.out.println();
//System.out.println();

//System.out.print("matB\n");
for (int l = 0; l<linha2; l++) {
    for (int c = 0; c<coluna2; c++) {
        matB[l][c] = (int)(Math.random()*100);
        //System.out.print(matB[l][c]+ " ");
    }
    //System.out.println();
}
//System.out.println();
//System.out.println();

System.out.println("Entre com a quantidade de threads: ");
numThreads = teclado.nextInt();

teclado.close();

//caso o numero de threads seja maior que o de linhas da matriz resultante
if(numThreads>linha1)
{
    numThreads=linha1;
}

//criando as threads
ParaleloDados[] threads = new ParaleloDados[numThreads];

for (int l = 0; l<numThreads; l++)
threads[l]= new ParaleloDados(linha1,coluna1,linha2,coluna2,matA,matB,matresp);

tempo1 = System.nanoTime();

//definindo o intervalo em cada thread vai executar
int divisao=linha1/numThreads,restdivisao=linha1%numThreads;

for(int j = 0; j < numThreads ; j++) {
    if((restdivisao)==0)
        threads[j].DefineIntervalo((divisao)*j, (divisao)*(j+1));
    else
    {
        if(j==numThreads-1)
```



## MINISTÉRIO DA EDUCAÇÃO

Universidade Federal de Itajubá - Campus Itabira  
Rua Irmã Ivone Drummond, 200 - Distrito Industrial II  
Itabira - MG - 35903-087

```
        threads[j].DefineIntervalo((divisao)*j,
(((divisao)*(j+1)))+(restdivisao)));
        else
        threads[j].DefineIntervalo((divisao)*j, (divisao)*(j+1));
    }

}

//inicia as threads
for(int j = 0; j < numThreads; j++)
    threads[j].start();

//define um ponto de sincronismo (barreira)
//aguarda o termino do processamento de todas as threads
for(int j = 0; j < numThreads; j++)
    threads[j].join();

tempo2 = System.nanoTime();

// System.out.print("Matriz Resultante\n");
// for (int l = 0; l<linha1; l++) {
//     for (int c = 0; c<coluna2; c++) {
//         System.out.print(matresp[l][c]+ " ");
//     }
//     System.out.println();
// }

    System.out.println("\tTempo = " + String.valueOf((tempo2 -
tempo1)/1000000) + " ms");

}

}
```