Aula de Revisão AV1

Desenvolvimento de Aplicações Distribuídas

Prof Anderson

CONTEÚDO PROGRAMÁTICO

UNIDADE 1 Introdução

- 1.1 Histórico
- 1.2 Conceitos básicos
- 1.3 Aspectos de Projeto (Transparência, Flexibilidade, Confiabilidade, Desempenho e Escalabilidade)

UNIDADE 2 Programação Concorrente

- 2.1 Ambiente de Programação
- 2.2 Objetivos
- 2.3 Java Threads

UNIDADE 3 Comunicação nos Sistemas Distribuídos

- 3.1 Modelo Cliente-Servidor
- 3.2 Serialização de Objetos em Java
- 3.3 Sockets, RPC e RMI

UNIDADE 4 Computação Distribuída

- 4.1 Conceitos
- 4.2 Modelo de Computação Distribuída
- 4.3 Web Services

INTRODUÇÃO

GRADE COMPUTACIONAL

Cluster	Grade Computacional
Solucionar problemas da organização.	Disponibilizar recursos e serviços de uma forma **geograficamente distribuída**
Gerenciamento por uma **autoridade central**	Cada **organização virtual gerencia** seus recursos mantendo a visão única do sistema
Voltado a performance e disponibilidade	Além disso prioridade para prestação de serviços
Custo para a organização	Custo dissolvido entre as organizações
Devido a natureza das aplicações não permite latência na comunicação	Permite latência devido as aplicações paralelas possuírem pouca ou nenhuma comunicação entre as tasks

INTRODUÇÃO

Aspectos de Projeto (Transparência, Flexibilidade, Confiabilidade, Desempenho e Escalabilidade)

Questão 3

Com relação às grades computacionais, marque a alternativa que apresenta os termos que completam a afirmativa:

“_____recursos e serviços de uma forma _____”.

a)Disponibilizar – geograficamente distribuída

b) Disponibilizar – localmente distribuído

c)Concentrar – geograficamente distribuída

d)Concentrar – localmente distribuída

e)Certos – única.

INTRODUÇÃO

Aspectos de Projeto (Transparência, Flexibilidade, Confiabilidade, Desempenho e Escalabilidade)

Questão 3

Com relação às grades computacionais, marque a alternativa que apresenta os termos que completam a afirmativa:

“_____recursos e serviços de uma forma _____”.

a)Disponibilizar – geograficamente distribuída

b) Disponibilizar – localmente distribuído

c)Concentrar – geograficamente distribuída

d)Concentrar – localmente distribuída

e)Certos – única.

INTRODUÇÃO

Aspectos de Projeto (Transparência, Flexibilidade, Confiabilidade, Desempenho e Escalabilidade)

Questão 4

Marque a alternativa que apresenta a definição para o termo CLUSTER

a)Dois ou mais computadores que funcionam descoordenados.

b)Dois ou mais computadores que não funcionam coordenados.

c)Um ou mais computadores que funcionam coordenados, como se fosse somente um.

d)Dois ou mais computadores que funcionam coordenados, como se fosse somente um.

e)Sistemas complexos que funcionam de forma complexa.

INTRODUÇÃO

Aspectos de Projeto (Transparência, Flexibilidade, Confiabilidade, Desempenho e Escalabilidade)

Questão 4

Marque a alternativa que apresenta a definição para o termo CLUSTER

a)Dois ou mais computadores que funcionam descoordenados.

b)Dois ou mais computadores que não funcionam coordenados.

c)Um ou mais computadores que funcionam coordenados, como se fosse somente um.

d)Dois ou mais computadores que funcionam coordenados, como se fosse somente um.

e)Sistemas complexos que funcionam de forma complexa.

INTRODUÇÃO

Aspectos de Projeto (Transparência, Flexibilidade, Confiabilidade, Desempenho e Escalabilidade)

Questão 5

Marque a alternativa que apresenta os termos que completam a afirmativa:

“Em _____ o custo é somente de uma organização, enquanto que em _____ o custo é compartilhado”.

a)Um cluster – uma grade

b)Um cluster – um cluster

c)Uma grade – Uma grade

d)Uma grade – um cluster

INTRODUÇÃO

Aspectos de Projeto (Transparência, Flexibilidade, Confiabilidade, Desempenho e Escalabilidade)

Questão 5

Marque a alternativa que apresenta os termos que completam a afirmativa:

“Em _____ o custo é somente de uma organização, enquanto que em _____ o custo é compartilhado”.

a)Um cluster – uma grade

b)Um cluster – um cluster

c)Uma grade – Uma grade

d)Uma grade – um cluster

INTRODUÇÃO

TIPOS DE CLUSTER

ALTA DISPONIBILIDADE

BALANCEAMENTO DE CARGA

INTRODUÇÃO

ALGORITMOS BALANCEAMENTO DE CARGA

LEAST CONNECTIONS

ROUND ROBIN

WEIGHTED FAIR

INTRODUÇÃO

ALGORITMOS BALANCEAMENTO DE CARGA

Questão 2

Nos últimos anos a computação gráfica praticamente dominou o mercado cinematográfico. Para realizar estes cálculos, são necessários algoritmos complexos de cálculo de iluminação que, muitas vezes acessam o mesmo recurso simultaneamente. Considerando que está sendo usado um fluxo sequencial simples de execução para o cálculo da iluminação, marque a alternativa da tecnologia utilizada para o processamento dessa cena.

a) thread
b) processo
c) round robin
d) least connection
e) weighted fair

INTRODUÇÃO

ALGORITMOS BALANCEAMENTO DE CARGA

Questão 2

Nos últimos anos a computação gráfica praticamente dominou o mercado cinematográfico. Para realizar estes cálculos, são necessários algoritmos complexos de cálculo de iluminação que, muitas vezes acessam o mesmo recurso simultaneamente. Considerando que está sendo usado um fluxo sequencial simples de execução para o cálculo da iluminação, marque a alternativa da tecnologia utilizada para o processamento dessa cena.

a) thread

b) processo

c) round robin

d) least connection

e) weighted fair

INTRODUÇÃO

ALGORITMOS BALANCEAMENTO DE CARGA

Questão 6

Marque a alternativa que apresenta um tipo de cluster cuja característica é dispor o serviço de forma ininterrupta.

a)Alta disponibilidade
b)Balanceamento de Carga
c)Balanceamento de Recursos
d)Alto balanceamento
e)Balanceamento de Disponibilidade

INTRODUÇÃO

ALGORITMOS BALANCEAMENTO DE CARGA

Questão 6

Marque a alternativa que apresenta um tipo de cluster cuja característica é dispor o serviço de forma ininterrupta.

a)Alta disponibilidade

b)Balanceamento de Carga

c)Balanceamento de Recursos

d)Alto balanceamento

e)Balanceamento de Disponibilidade

INTRODUÇÃO

Some Computer Organizations and Their Effectiveness

MICHAEL J. FLYNN, MEMBER, IEEE

1) The single-instruction stream–single-data stream organization (SISD), which represents most conventional computing equipment available today.

2) The single-instruction stream–multiple-data stream (SIMD), which includes most array processes, including Solomon [2] (Illiac IV).

3) Multiple-instruction stream–single-data stream type organizations (MISD), which include specialized streaming organizations using multiple-instruction streams on a single sequence of data and the derivatives thereof. The plug-board machines of a bygone era were a degenerate form of MISD wherein the instruction streams were single instructions, and a derived datum (SD) passed from program step i to program step $i+1$ (MI).

4) Multiple-instruction stream–multiple-data stream (MIMD), which include organizations referred to as "multiprocessor." Univac [3], among other corporations, has proposed various MIMD structures.

INTRODUÇÃO

FLYNN

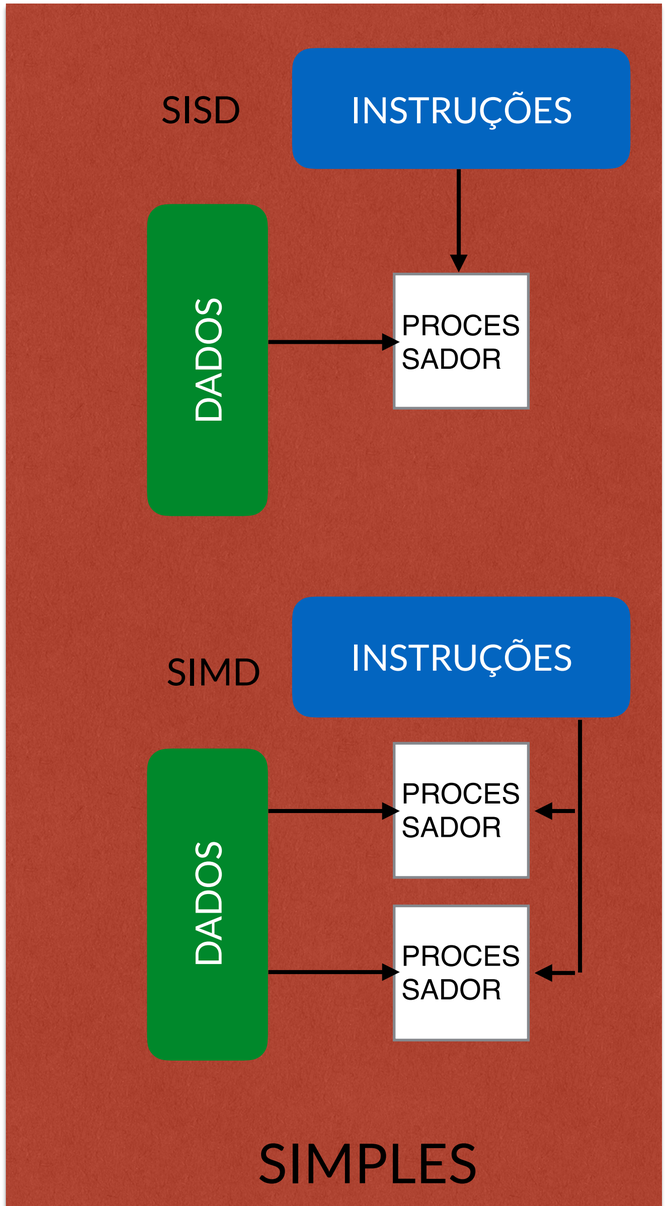
- Baseada nos fluxos de *instruções* e *dados*
- Produto:

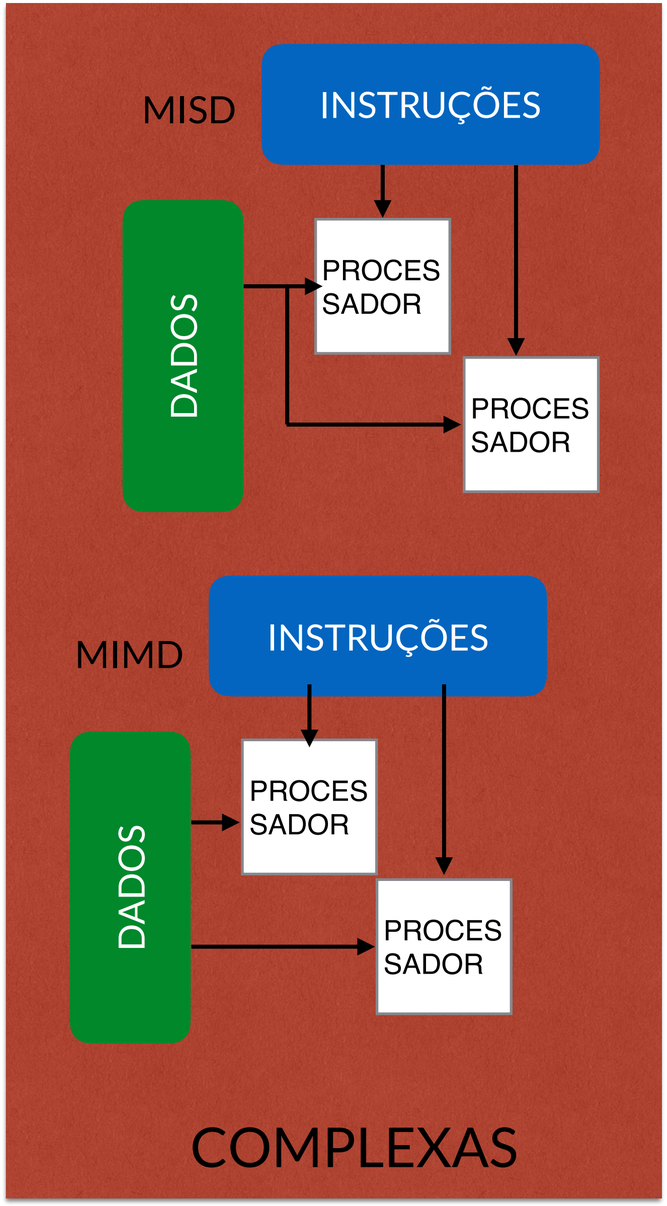
  {fluxos-de-instruções} x {fluxos-de-dados}
- Quatro casos possíveis:
  - SISD (Single-Instruction, Single-Data)
  - SIMD (Single-Instruction, Multiple-Data)
  - MISD (Multiple-Instruction, Single-Data)
  - MIMD (Multiple-Instruction, Multiple-Data)

INTRODUÇÃO

FLYNN

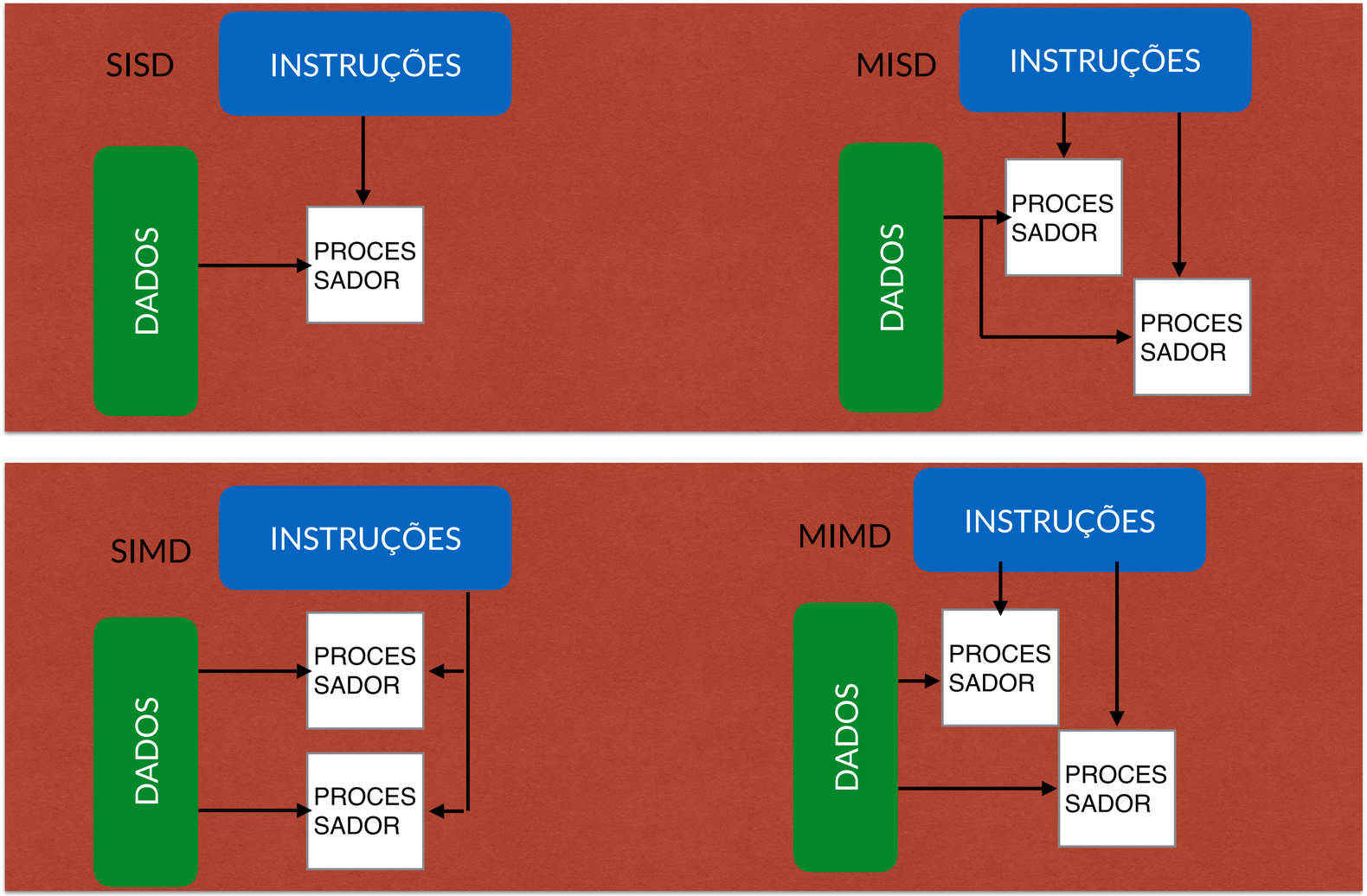
INSTRUÇÕES





INTRODUÇÃO

FLYNN



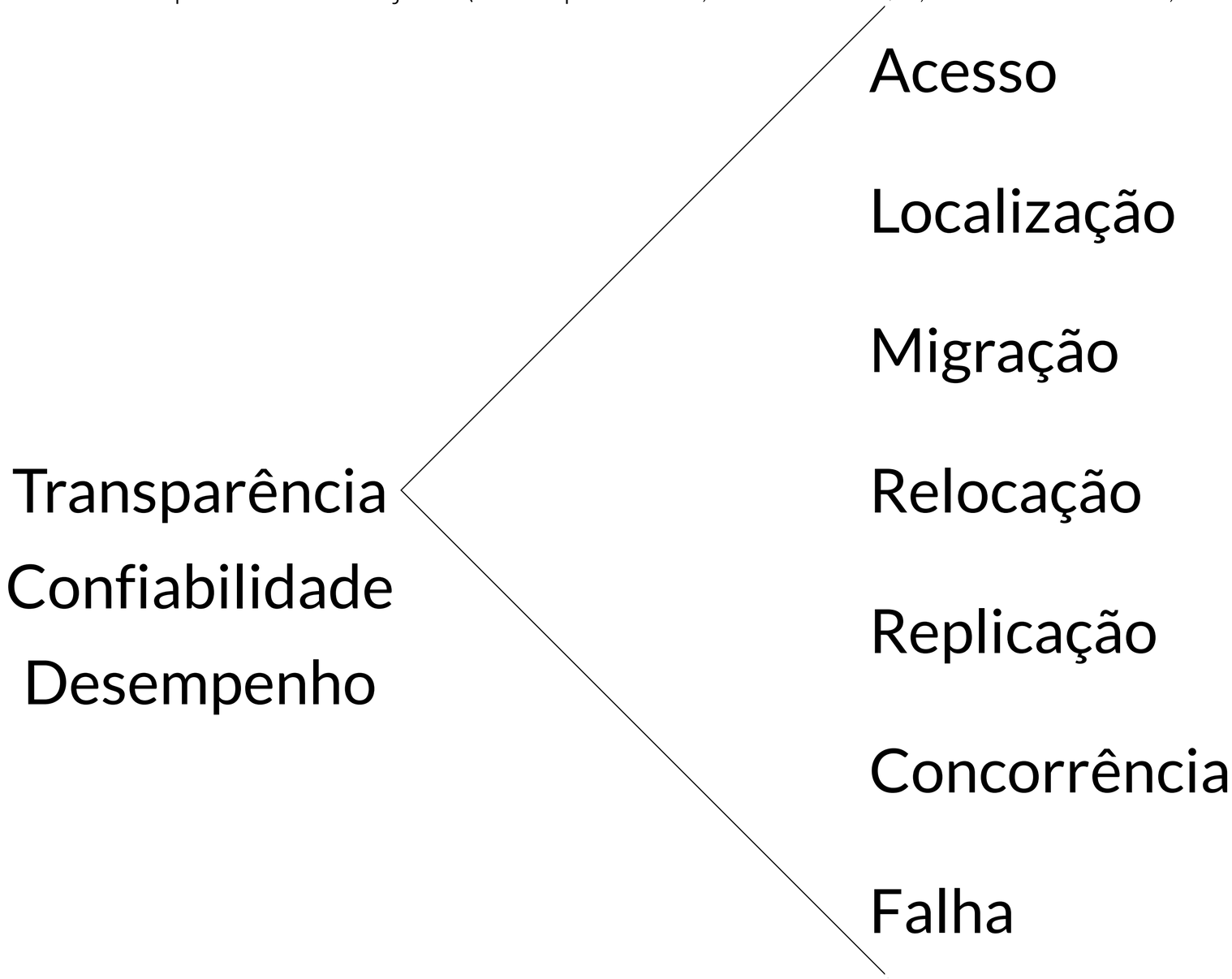
DADOS

SIMPLES

COMPLEXOS

INTRODUÇÃO

Aspectos de Projeto (Transparência, Flexibilidade, Confiabilidade, Desempenho e Escalabilidade)



INTRODUÇÃO

Aspectos de Projeto (Transparência, Flexibilidade, Confiabilidade, Desempenho e Escalabilidade)

Questão 1

Oculta que um recurso possa ser movido para outra localização, sem estar sendo usado. Este tipo de transparência é conhecido como:

a)Replicação
b)Relocação
c)Migração
d)Localização
e)Acesso

INTRODUÇÃO

Aspectos de Projeto (Transparência, Flexibilidade, Confiabilidade, Desempenho e Escalabilidade)

Questão 1

Oculta que um recurso possa ser movido para outra localização, sem estar sendo usado. Este tipo de transparência é conhecido como:

a)Replicação
b)Relocação
c)Migração
d)Localização
e)Acesso

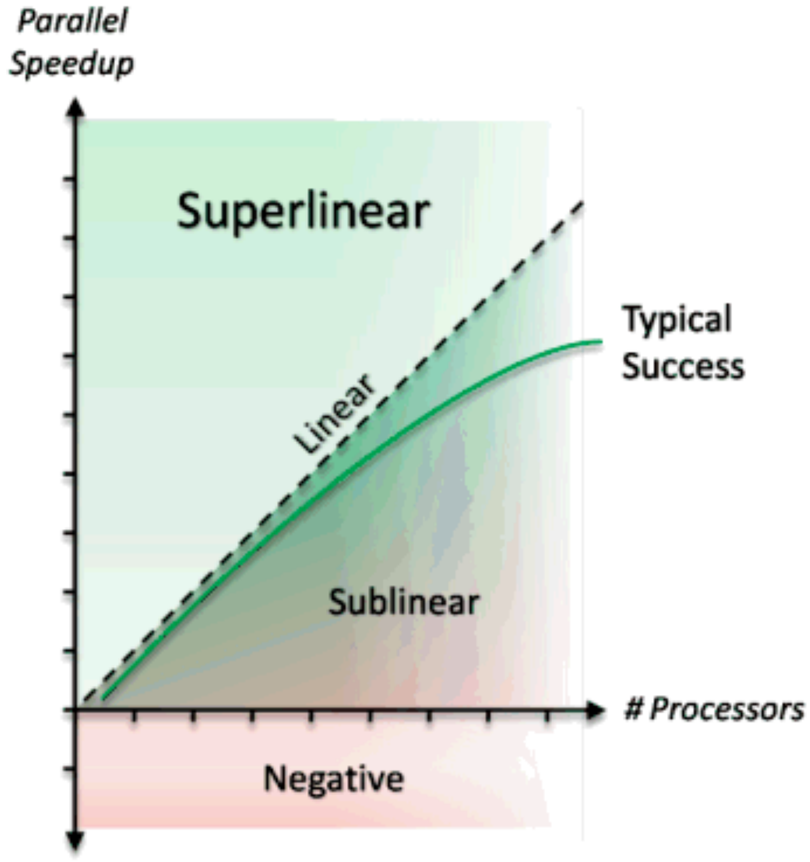
INTRODUÇÃO

Aspectos de Projeto (Transparência, Flexibilidade, Confiabilidade, Desempenho e Escalabilidade)

Desempenho: speedup

$$speedup = \frac{T_{seq}}{T_{par}}$$

Lei de Amdahl



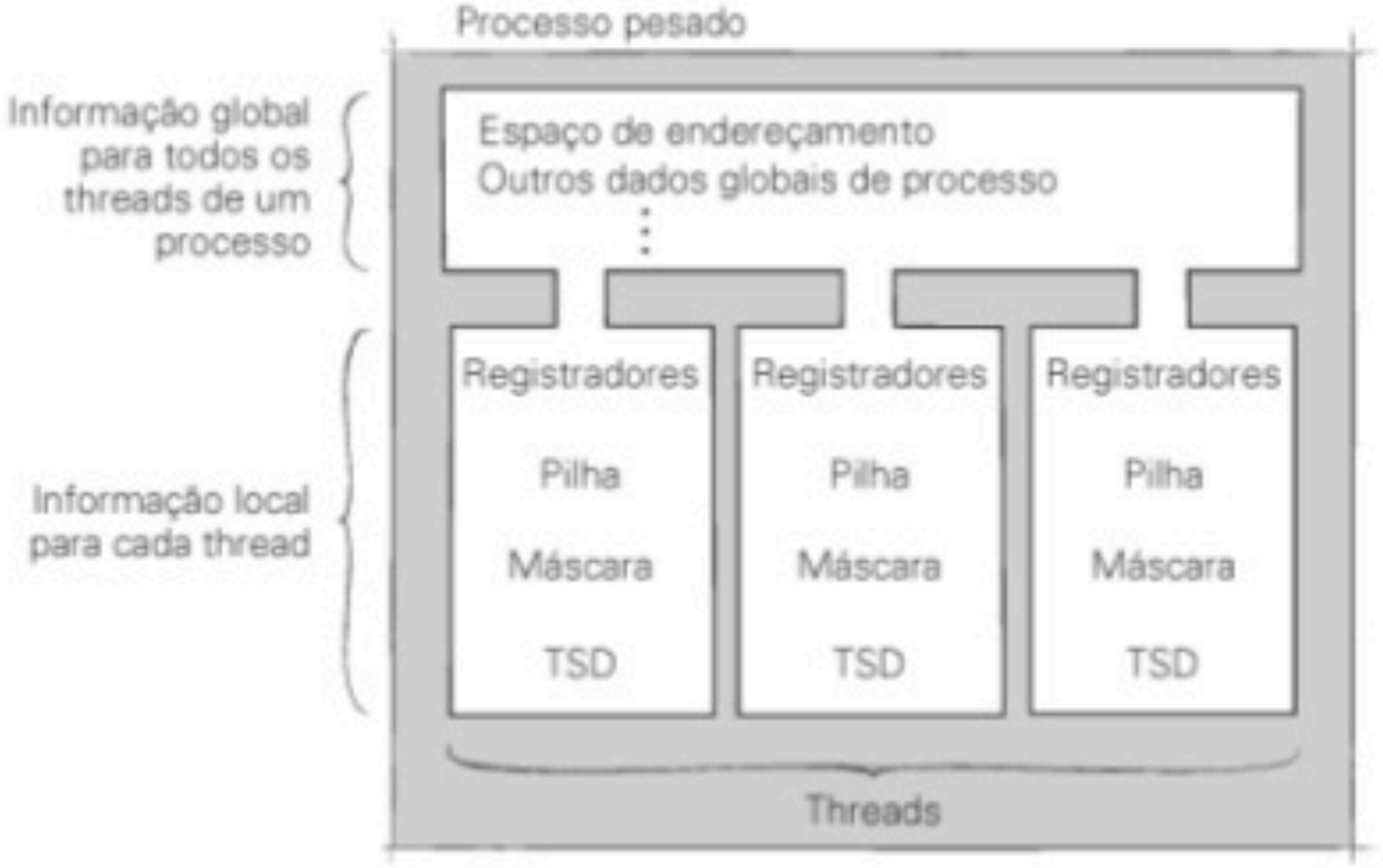
PROGRAMAÇÃO CONCORRENTE

INTRODUÇÃO

Thread

x

Processo



PROGRAMAÇÃO CONCORRENTE

INTRODUÇÃO

class Thread

interface Runnable

```
class PrimeiraRun implements Runnable {
        int i;
        PrimeiraRun(int i) {
            this.i = i;
        }

        public void run() {
            // faz alguma coisa…
        }
    }

class PrimeiraThread extends Thread {
        int i;
        PrimeiraThread(int i) {
            this.i = i;
        }

        public void run() {
            // faz alguma coisa…
        }
    }
```

PROGRAMAÇÃO CONCORRENTE

EXERCÍCIOS

```
public class Fibonacci extends Thread {

    private int n;

    public Fibonacci(int n) {
        this.n = n;
    }

    public int Calculo(int n) {
        …..
    }

    public void run() {
        for (int i = 0; i < n; i++) {
            System.out.println("Fibonacci:" + i + " " + Calculo(i));
            try {
                sleep((long) (Math.random() * 1000));
            } catch (InterruptedException e) {
            }
        }
        System.out.println("Fibonacci Terminou");
    }
}
```

```
public class Fibonacci implements Runnable {

    private int n;

    public Fibonacci(int n) {
        this.n = n;
    }

    public int Calculo(int n) {
        ….
    }

    public void run() {
        for (int i = 0; i < n; i++) {
            System.out.println("Fibonacci:" + i + " " + Calculo(i));
            try {
                sleep((long) (Math.random() * 1000));
            } catch (InterruptedException e) {
            }
        }
        System.out.println("Fibonacci Terminou");
    }
}
```

PROGRAMAÇÃO CONCORRENTE

EXERCÍCIOS

public class Fibonacci extends Thread {

```
    private int n;

    public Fibonacci(int n) {
        this.n = n;
    }

    public int Calculo(int n) {
        …..
    }

    public void run() {
        for (int i = 0; i < n; i++) {
            System.out.println("Fibonacci:" + i + " " + Calculo(i));
            try {
                sleep((long) (Math.random() * 1000));
            } catch (InterruptedException e) {
            }
        }
        System.out.println("Fibonacci Terminou");
    }
}
```

public class Fibonacci implements Runnable {

```
    private int n;

    public Fibonacci(int n) {
        this.n = n;
    }

    public int Calculo(int n) {
        ….
    }

    public void run() {
        for (int i = 0; i < n; i++) {
            System.out.println("Fibonacci:" + i + " " + Calculo(i));
            try {
                sleep((long) (Math.random() * 1000));
            } catch (InterruptedException e) {
            }
        }
        System.out.println("Fibonacci Terminou");
    }
}
```

PROGRAMAÇÃO CONCORRENTE

EXERCÍCIOS

```
public class ExemploThread {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        (new Fatorial(10)).start();
        (new Fibonacci(10)).start();
    }

}
```

```
public class ExemploRunnable {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        (new Thread(new Fibonacci(10))).start();
        (new Thread(new Fatorial(10))).start();
    }

}
```


OBRIGADO!
Boa-prova!