Seminário Final

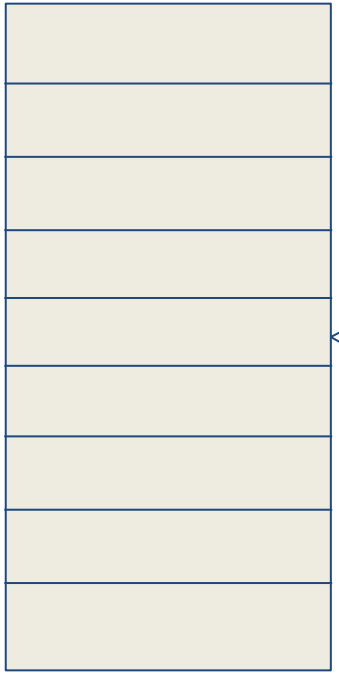
Disciplina de Sistemas Operacionais

Marina Silva

Objetivo

- Implementar em Java um algoritmo paralelo/concorrente para gerenciamento dinâmico de memória.
    - Alocação de memória (heap);
    - Vetor de requisições (fila circular);
    - Algoritmo para gerenciar alocação e desalocação de memória em paralelo.

Heap



- Vetor de Blocos (Páginas)

```
public class Bloco {
        private Integer id;
        private Integer acesso;
        private Semaphore semaforo;
}
```

```java
public void alocarVariavel(VetorRequisicoes requisicoes) {
    Requisicao requisicao = requisicoes.remover();
    Integer tamanhoVariavel = requisicao.getTamVariavel();
    //altera a variavel alocado
    this.setAlocado(this.getAlocado() + (tamanhoVariavel * tamanhoBloco));
    Integer instante = gerarInstante();
    //algoritmo de alocação
    for (int i = 0; i < vetor.length; i++) {
        if (vetor[i].getId() == 0 && tamanhoVariavel > 0) {
            Bloco alocar = new Bloco(requisicao.getId(), instante);
            try {
                vetor[i].getSemaforo().acquire();
                vetor[i] = alocar;
            } catch (InterruptedException ex) {
                Logger.getLogger(Heap.class.getName()).log(Level.SEVERE, null, ex);
            } finally {
                vetor[i].getSemaforo().release();
            }
            tamanhoVariavel--;
        }
```

Pagina{id=0, acesso=0}
Pagina{id=0, acesso=0}
Pagina{id=32, acesso=0}
Pagina{id=32, acesso=0}
Pagina{id=774, acesso=0}
Pagina{id=774, acesso=0}
Pagina{id=774, acesso=0}
Pagina{id=32, acesso=0}
Pagina{id=32, acesso=0}
Pagina{id=32, acesso=0}
Pagina{id=32, acesso=0}

```java
public void desalocarVariavel() {
    Integer aloc = this.getAlocado() / this.getTamanhoBloco();
    Bloco vazia = new Bloco(0, 0);
    //algoritmo de desalocação LRU
    for (int i = 0; i < (vetor.length); i++) {
        if (vetor[i].getAcesso() == 0) {
            try {
                //System.out.println("desaloca posicao: " + i);
                vetor[i].getSemaforo().acquire();
                vetor[i] = vazia;
                aloc--;
            } catch (InterruptedException ex) {
                Logger.getLogger(Heap.class.getName()).log(Level.SEVERE, null, ex);
            } finally {
                vetor[i].getSemaforo().release();
            }
        }
    }

    this.setAlocado(aloc * this.getTamanhoBloco());
```

```
//paralelo
ExecutorService executor = Executors.newFixedThreadPool(2);
long tempoInicial = System.currentTimeMillis();
do {
    executor.execute(alocar);
    requisicoes--;

    if (heap.getAlocado() >= conf.getLimiarMaximoHeap()) {
        executor.execute(desalocar);
    }
} while (requisicoes != 0);
executor.shutdown();
System.out.println("Paralelo: " + (System.currentTimeMillis() - tempoInicial) + "ms"
```

Testes

- Linux Mint 19.1
- Intel© Core™ i5-4200U CPU @ 1.60GHz
- Memória RAM 4 Gb

Heap: 512b Página: 8b 1000 req.

- Sequencial: 75 ms
- Paralelo: 9,5 ms

Heap: 1024b Página: 8b 1000 req.

- Sequencial: 75,4 ms
- Paralelo: 16 ms

Heap: 1024b Página: 16b 10000 req.

- Sequencial: 915,8ms
- Paralelo: 28,6 ms

Obrigada!