

PROVA I

NOME: JHONAT HEBENSON AVELINO DE SENA
MATRÍCULA: 2020 00~~00~~ 580

- * CAPÍTULO 1: 1-6, 9 (7 QUESTÕES)
- * CAPÍTULO 2: 1-3, 5, 7, 15-16, 19-21, 24 (12 QUESTÕES)
- * CAPÍTULO 5: 6, 8-9, 12-14 (6 QUESTÕES)
- * PROGRAMAÇÃO ASSINADA: 1-4 (4 QUESTÕES)

TOTAL: 29 DE 31

CAPITULO I

Q1) DEVISE formulas for the functions that calculate my first "i" and my last "i" in the global sum Examples. REMEMBER THAT EACH CASE SHOULD BE ASSIGNED ROUGHLY THE SAME NUMBER OF COMPUTATIONS IN THE LOOP. HINT: FIRST CONSIDER THE CASE WHEN n IS EVENLY DIVISIBLE BY P

$$I = [0 \ 1 \ 2]$$

$$n = 12 \quad \frac{n}{P} = 3 \leftarrow 3 \text{ divisions}$$

$$P = 4$$

$$[0 \quad 4 \quad 8]$$

$$[3 \quad 7 \quad 11 \quad 12]$$

Tenor o SEGUNTE Formula

$$\text{MY-FIRST-I}(i) = i \cdot P$$

$$\text{MY-LAST-I}(i) = i \cdot P + \frac{n}{P}$$

* PARA QUANDO n ñ ë DIVISÍVEL

Tenor

$$\begin{matrix} n = 14 \\ P = 4 \end{matrix}$$

$$\frac{n}{P} = 3,5$$

$$[0 \quad 4 \quad 8 \quad 12 \quad 13]$$

$$I = [0 \ 1 \ 2 \ 3]$$

$$[3 \quad 7 \quad 11 \quad 13]$$

$$REST = N \times P$$

$$\text{If } (i > REST) \quad \sum_{j=i}^{N-1} \frac{P}{P} = i \times \frac{P}{P} + REST + I$$

$$\left\{ \begin{array}{l} \\ \\ MY - FIRST - I = MY - LAST - I + \frac{P}{P} = I \end{array} \right.$$

1.2) PRESUMIMOS IMPLICITAMENTE QUE CADA CHAMADA PARA COMPUTADOR REQUEIRE UM VALOR APROXIMO A MESMA QUANTIDADE DE TRABALHO que AS OUTRAS CHAMADAS.

COMO VOCÊ MUDARIA SUA RESPOSTA à PERGUNTA ANTERIOR SE A CHAMADA $i=k$ REQUER $k+1$ VEZES MAIS TRABALHAR com A CHAMADA COM $i=0$? PORTANTO, SE A PRIMEIRA CHAMADA ($i=0$) REQUER 2 milisegundos a SEGUNDA CHAMADA ($i=1$) REQUER 4, a TERCEIRA ($i=2$) REQUER 6 ms por diante

Como o CUSTO de cada requisição é:

$$f(n, 2, i)$$

LOGO NOS POSSIVEL EXTRAPOLAR SERIA QUE CADA NÚCLEO FICARIA COM INSTÂNCIAS DOS PROCESSOS RESTANTES:

$$\text{EX: } n = 12$$

$$i = [0, 1, 2]$$

$$P = 4$$

ANTES SEDIA:

CUSTOS: 2 4 6 8 10 12 14 16

$$18 \quad 20 \quad 22 \quad 24 = 756$$

Com ESSA NOVA ABORDAGEM:

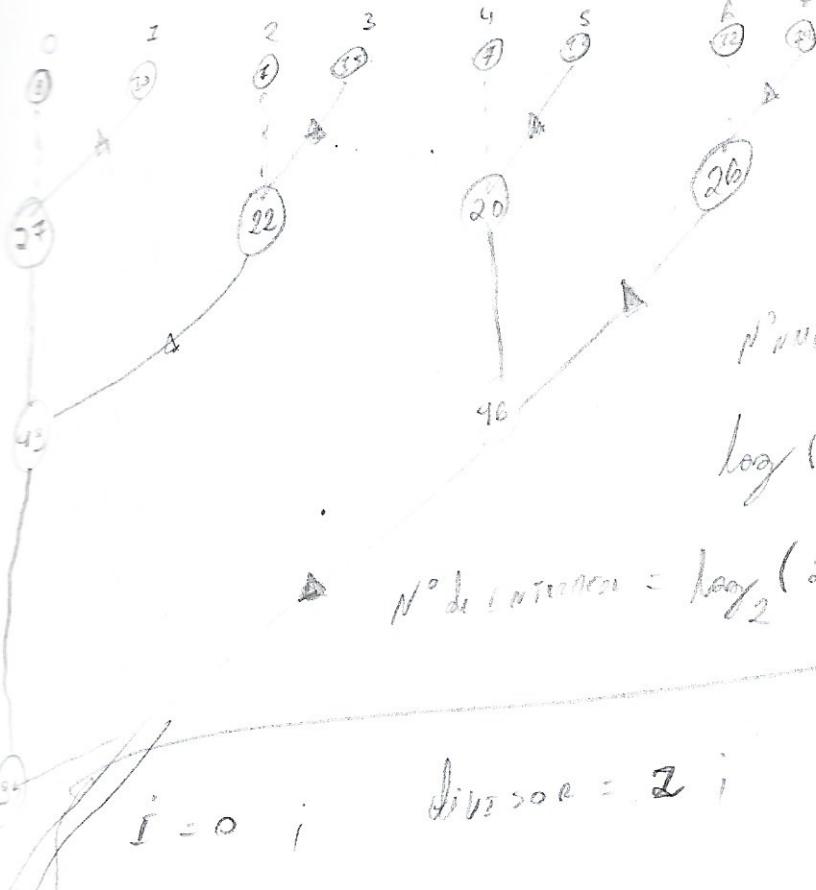
$$\text{NÚCLEO 0} \rightarrow [0, 1, 2, 3, 4, 5] \rightarrow \text{CUSTO } [2, 4, 6, 8, 10]$$

$$\text{NÚCLEO 1} \rightarrow [6, 7, 8] \rightarrow \text{CUSTO} = 92 + 32 = 54$$

$$\text{NÚCLEO 2} \rightarrow [5, 10, 11] \rightarrow \text{CUSTO} = 54 + 72 = 66$$

1.3) TENTE ESCREVER UM PSEUDO-CÓDIGO
PARA A SOMA GLOBAL ESTRUTURADA
EM ÁRVORE ILUSTRADA EM
FIGURA I.1 SUPONHA QUE O
NÚMERO DE NÚCLEOS SEJA POTÊNCIA
DE 2 (1, 2, 4, 8...).

DECA: USE UM DIVISOR DE VARIAVEL
PARA DETERMINAR SE UM NÚCLEO DEVE
ENVIAR SENO SOMAR OU RECEBER E
ADICIONAR. O DIVISOR DEVE COMEÇAR
COM O VALOR 2 E SER DOBRADO
APÓS CADA ITERAÇÃO. USE TAMBÉM
UMA DIFERENÇA DE NÚCLEO VARIAVEL
PARA DETERMINAR QUAL NÚCLEO DEVE
SER ASSOCIADO AO NÚCLEO ATUAL.
DEVE COMEÇAR COM O VALOR I
E TAMBÉM SER DUPLICADO APÓS CADA
ITERAÇÃO. Por exemplo em A
PRIMEIRA ITERAÇÃO. O divisor = 0 e
I = divisor = 1 entra o RECEBER E
ADICIONAR, ENQUANTO I ENVIAR.
TAMBÉM NA PRIMEIRA ITERAÇÃO
O + DIFERENÇA DE NÚCLEO = 1 E
I - DIFERENÇA DE NÚCLEO = 0 ENTÃO O
I E I SÓ PAREADOS NA PRIMEIRA
ITERAÇÃO.



$$\log_2(8) = 3$$

$$n^{\text{nucl}} = 2^n$$

$$\log(n^{\text{nucl}})$$

$$\Rightarrow n^{\text{division}} = \log_2(2^n) \Theta$$

~~2 + (i = 0; i < log2(p); i++) {~~ DIFERENCA - NUCLEO = 0

~~i = 0; i < log2(p); i++) {~~

FOR ~~(i = 0; i < log2(p); i++) {~~ }
~~R = i + 2~~
~~i & (i - cont * divisor == 0) {~~

RECIBE E ADICIONA DE
 R

(I - cont + DIFERENCA - NUCLEO)
 (I - cont + DIFERENCA - NUCLEO)

} ISE E
 ENVIAR PARA [I - cont + DIFERENCA - NUCLEO]

}

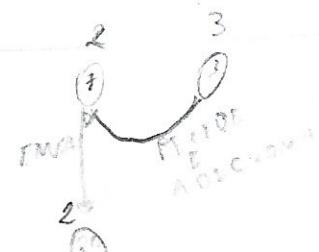
DIVISOR = DIVISOR * 2

DIFERENCA - NUCLEO = DIFERENCA - NUCLEO * 2

EX: ~~I - cont = 2, 3~~
~~i = 0 divisor = 2~~
~~DIFERENCA - NUCLEO = 1~~

$$2 \times 2 = 0$$

$$3 \times 2 \neq 0$$



$$i = 3$$

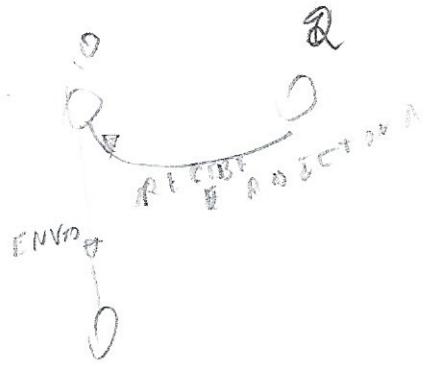
$$\text{diferencia} = 4$$

$$\text{diferencia - numero} = 2$$

$$I_{\text{cont}} = 0 + \frac{q}{R}$$

$$\begin{aligned} 0 \% .4 &= 0 \\ 2 \% .4 &\neq 0 \end{aligned}$$

$$\begin{aligned} &\text{P1} \text{ P2} \\ &\text{P1} \text{ P2} \\ &0 \times 2 \end{aligned}$$



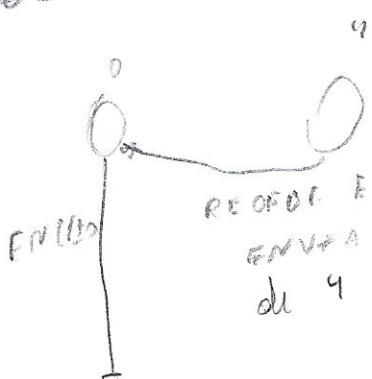
$$i = 2$$

$$\text{diferencia} = 3$$

$$I_{\text{cont}} = 0,4 \quad \text{diferencia - numero} = 4$$

$$0 \% .8 = 0$$

$$4 \% .8 \neq 0$$



$$I_{\text{cont}} = 0 + i \cdot 2$$

$$i = 3 \quad R = 1$$

$$i = 4 \quad R = 2$$

$$i = 2 \quad R = 4$$

$$R = \frac{R}{2}$$

PA

$$I_{\text{cont}} = 0 + (i-1) \cdot R$$

I. 4) COMO ALTERNATIVA À ABORDA GEN

DESCRITA NO PROBLEMA ANTERIOR, PODEMOS USAR OS OPERADORES \oplus & $\oplus\lnot$ DE C PARA IMPLEMENTAR A SOMA GLOBAL ESTRUTURADA EM ÁRVORE. A FIM DE VEDA COMO ISSO FUNCIONA, AJUDA A ESCREVER A REPRESENTAÇÃO BINÁRIA (BASE 2) DE CADA UMA DAS CLASSESI CASA COM PRINCIPAIS E OBSERVE OS PARES DURANTE CADA ESTAGIO:

Cores	STAGES		
	1	2	3
$D_{10} = 00_2$	$1_{10} = 00_2$	$2_{10} = 01_2$	$4_{10} = 10_2$
$1_{10} = 00_2$	$0_{10} = 00_2$	X	X
$2_{10} = 01_2$	$3_{10} = 01_2$	$0_{10} = 00_2$	$0_{10} = 00_2$
$3_{10} =$		X	X
.	.	.	.
.	?	?	?

A PARTIR DA TABELA VEMOS QUE DURANTE A PRIMEIRA FASE, CADA NÚCLEO É EMPARELHADO COM O NÚCLEO CUIJA CLASSE FECHAÇÃO DIFERE NO BIT MAIS À DIREITA DO PRIMEIRO. DURANTE A SEGUNDA FASE OS PRIMEIROS CONTINHAM SÓS PAREADOS NÚCLEOS QUE CONTINHAM SÓS PAREADOS COM O NÚCLEO CUIJA CLASSE FECHAÇÃO DIFERE ~~NÚCLEOS~~ CUIJA NO SÉGUNDO BIT E DURANTE O TERCEIRO ESTÁGIO OS NÚCLEOS SÓS EMPARELHADOS COM O NÚCLEO CUIJA CLASSE FECHAÇÃO

DIFERE NO TERCEIRO BIT

ASSIM, SE TIVÉRMOS UMA MÁSCARA DE
BITS DE VALOR BINÁRIO QUE É

0 0 1 2 PARA O PRIMEIRO ESTÁGIO

0 0 0 2 PARA O SEGUNDO E 1 0 0 2 PARA

O TERCEIRO PODEROS DÁ-LO A CLASSIFICAÇÃO

DE NÚCLEO COM O QUAL ESTAMOS

EMPARAELHADOS "INVERTENDO" A PARTE EM

NOSSA CLASSIFICAÇÃO QUE É DIFERENTE

DE ZERO EM BITMASK. ISSO PODE

SER FEITO USANDO O OPERADOR BIT A BIT

EXCLUSIVO OU A IMPLEMENTE ESTE

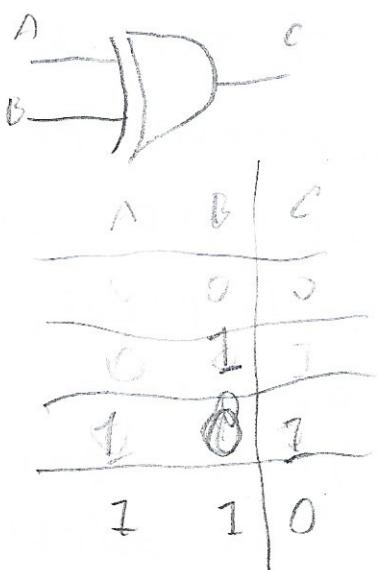
ALGORITMO EM PSEUDO-CÓDIGO

USANDO O BIT A BIT EXCLUSIVO

ON E OPERAÇÃO DE TORNO ESTENDIDA.

$$\begin{array}{r} 0 \ 0 \ 1 \\ 0 \ 0 \ 1 \\ \hline 0 \ 1 \ 2 \end{array}$$

3



IF (ESTACAO == 2) {
 I - NUCLEO - PARDAO[2] = I - ATUAL[2];
 I - NUCLEO - PARDAO[1] = I - ATUAL[1];
 I - NUCLEO - PARDAO[0] = I;
 IF (I - ATUAL[0] & 1) {
 I = I - NUCLEO - PARDAO[0];
 } ELSE {
 I = NUCLEO - PARDAO[0];
 }
 }

}

 }

 }

 IF (ESTACAO == 2) {
 I - NUCLEO - PARDAO[0] = I - ATUAL[0];
 I - NUCLEO - PARDAO[2] = I - ATUAL[2];
 I & (I - ATUAL & 7) {
 I = NUCLEO - PARDAO[7];
 } ELSE {
 I = NUCLEO - PARDAO[7];
 }
 }

 }

 }

 IF (ESTACAO == 3) {
 I - NUCLEO - PARDAO[1] = I - ATUAL[3];
 I - NUCLEO - PARDAO[0] = I - ATUAL[0];
 IF (I - ATUAL[2] & 1) {

I - NUCLEO - PARDAO[2] = I;

BELSE E
I - NUCLEO - PAREO = 0)

}

}

1.5) O QUEM ACONTECE SE O SEU PRÉNODO CÓDIGO

NO EXERCÍCIO 1.4 FOR EXECUTADO

QUANDO O NÚMERO DE NÚCLEOS

NÃO É UMA POTÊNCIA DE DOIS (3, 5, 67)?

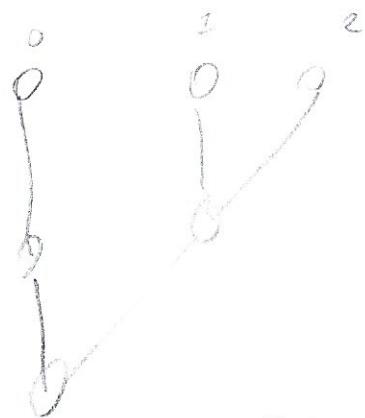
JOSE PODER NOTIFICAR O PSIUDO-CÓDIGO

PARA QUE FUNCIONE corretamente,

DEPENDENTEMENTE DO NÚMERO DE

NÚCLEOS

EX: 1.3



$$I_{-cont} = 0, I_{-2}.$$

$$i=0 \quad \text{dilino} = 2 \quad 0 \neq 2 \\ 1 \neq 2 \neq 0$$

$$\text{DESENHA-NÚCLEO} = 2$$

$$2+1=3$$

No desenho

EX: 1.4

No se FUNCIONAR

porque no CONSEGUE SER

PAREADOS, en qual qmco corpo

NA soma, con 6 NUCLEOS

CONSEGUE EMPAETAS N^1

PRIMERA, Poco M SISTEMA

No poden dables $\boxed{3/2 = 2,5}$

FORMA DE NMDOAN \rightarrow ALGORITMO I - 3

DISTRIBUIR A PARCETA QM SOBRA.

PARO nm con DESPERDICIO, ASIM CONSEGUNDA

QAN o PERMITIR. EXEMPLO

DIVISION = 2 DIFFERENCIA - NUCLEO = 1

N^o cores = cont - SYSTEM

For ($i = 0 \dots n - 1$) {

IF ($I - cont \neq 0$) division == 0 $\&$ $I - cont + 1 \neq$

N^o cores) {

RECIBE I soma de N^o cores DE

I - cont + 1 NUCLEO;

DE $\{ N^o$ cores $\neq 1 \} = 0 \& I - cont = N^o$ cores - 1 } $\}$

RECIBE I soma DE N^o cores = 1

3 ELSE {

$$\text{IF } \{n_cores \% 2\} = 0 \quad \text{DO } I_core + 1 = n_cores \}$$

ENVIA PARA I-core - DIFERENCA = NUCLEI-2

}

ELSE {

ENVIA PARA I-core - DIFERENCA - NUCLEO

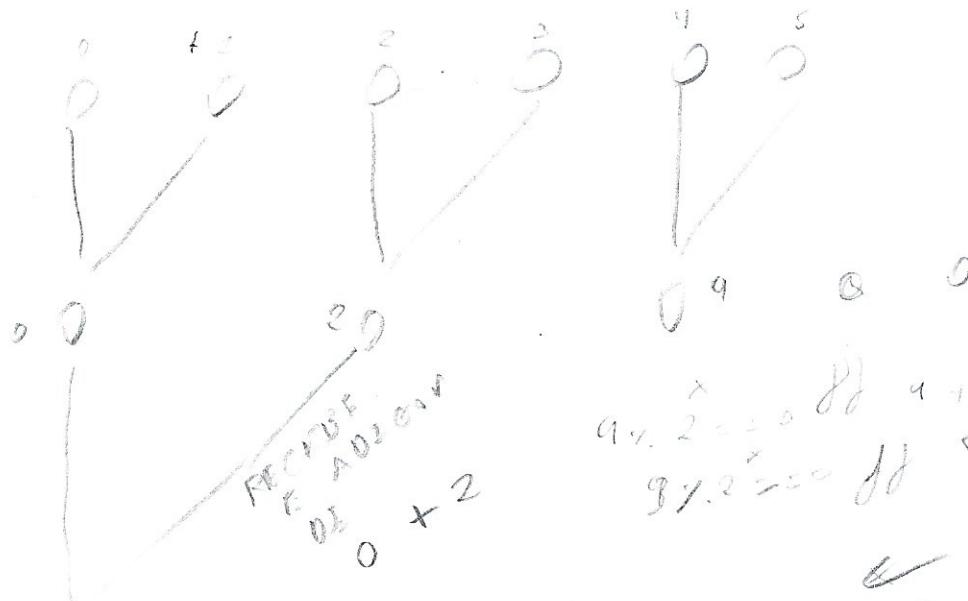
}

3

DEVISOR = DIVISOR * 2

DIFERENCA - NUCLEO = DISTANCIA - NUCLEO * 2

N_NUCLEOS = 1 N_NUCLEOS = 2ⁱ



$$9 \times 2 = 18 \quad \text{DO } 9 + 2 = 3 \quad \checkmark$$

$$9 \times 2 = 18 \quad \text{DO } 9 - 2 = 7 \quad \checkmark$$

$$I_core = 0, 2, 0$$

$$N_NUCLEOS = 6^4$$

$$DIVISOR = 4$$

$$DISTANCIA - NUCLEOS = 2$$

$$6 \quad \text{DO } 6 + 2 = 3 \quad \checkmark$$

$$6 \times 2 = 12 \quad \text{DO } 6 - 2 = 4 \quad \times$$

$$3 \times 2 = 6 \quad \text{DO } 6 - 2 = 4 \quad \checkmark$$

Q.6) DERIVE FóRMULAS PARA O
NÚMERO DE RECEBIMENTOS
E ADICÕES ANTES DE NMCLÉO
O REALIZA MUNDOS:

a) O ISENDO CÓDIGO ORIGINAIS PARA
uma soma GLOBAL?

b) soma GLOBAL ESTRUTURADA EM
ARVORE?

d)

$$\text{RECEBIMENTOS} = P - I$$

$$\text{ADICÕES} = \frac{n}{P} + (P - I)$$

~~RECEBIMENTOS = ADICÕES = (n - I)~~

n é NÚMEROS DE PROCESSAMENTO

Porque o NMCLÉO "0" RECEBEIA
o RESULTADO DE CADA MÚLTIOS
NMCLÉOS E SOMA

b)

$$RECEBIMENTOS = \frac{P_+}{2} - I = ADICAS^5$$

$$\cancel{ADICAS^5} = P$$

ON

$$RECEBEDOS = ADICAS^5 = 1\% (P)$$

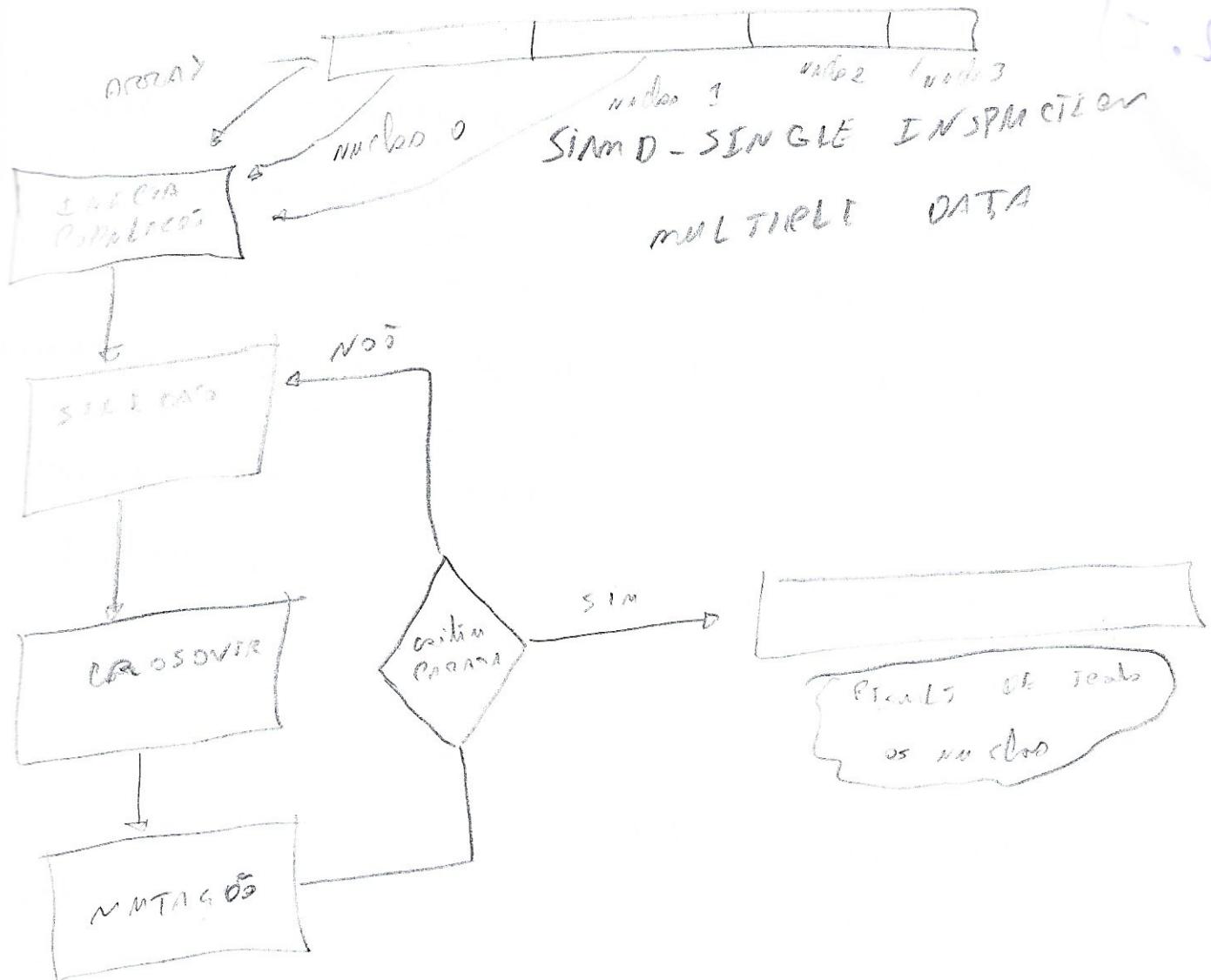
O NÚMERO "0" RECEBIDA \rightarrow VALOR DAS

SAMAS PARCIAIS E SOMADA, 6000

$$RECEBEDOS = ADICAS^5$$

I.3) ESCRIVA um ENSAIO DESCRITIVO
um PROBLEMA DE PESQUISA Em
UMA ÁREA DE ESPECIALIZAÇÃO
DO USO DE COMPUTADORES
PARALELA, FORNEÇA UM ESBOÇO DE
COMO ~~PARALELO~~ O PARALELISMO
SERIA USADO VOCÊ
USARIA PARALELISMO DE
TAREFAS ou DADOS?

NA MÍNHA ÁREA DE INTELLIGÊNCIA
ARTIFICIAL, POR EXEMPLO o
ALGORITMO GENÉTICO, UMA
BOA TÉCNICA SERIA UTILIZADA.
PARALELISMO DE DADOS, DE VEJO
O PARALELISMO DE TAREFA
NÓ SÉ JA MAIS BOA ABORDAGEM
PARA ESSE ALGORITMO.



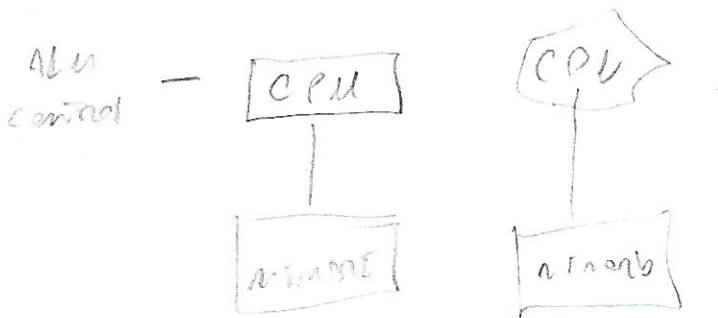
A IDEA E DIVIDIR A POPULAÇÃO
 INTEIRAMENTE em partes e espes
 partes para processadas m
 digraváriam cada uma das subidas.

2. I)

CAPÍTULO 2

QUANDO ESTAVAMOS DISCUTINDO A ADEGA
DE PONTO FLUTUANTE SIMPLESSEMOS
PRESSUPONDENDO QUE CADA UMA DAS UNIDADES
FUNCIONAIS LEVOU A MESMA QUANTIDADE
DE TEMPO. SUPONHA QUE CADA BUSCA
E ARMAZENAMENTO LEVA 2 NANOSEGUNDOS
E RESTANTE CADA OPERAÇÃO LEVA 1
NANOSEGUNDO.

a) QUANTO TEMPO LEVA UM
ADEGA DE PONTO FLUTUANTE COM
ESSAS SUPosições?



* PARA ADEGAS DE PONTO FLUTUANTE PRECISAMOS
DE P.º OPERAÇÃO [BUSCAR OPERANDOS, COMPARAR INCONTO,
MONTAR UM OPERANDO, ADICIONAR, NORMALIZAR O
RESULTADO, ARREDONDAR. O RESULTADO, ARMazenar o
RESULTADO.

$$2 + 4 + 1 + 1 + 1 + 1 + 2 = 11 \text{ nS}$$

2.I b) Quanto tempo LEVANA uns ADICAO
SEM PEPERLINE DE 1.000 PARES DE FLUTUADORES
com ISSAS SUBSEGUN ?

Com PEPERLINE polimer PARALIZADA ISSA SARA
COMO ESTAMOS SEM UMA ADICAO LEVA 9 NO
QUEREMOS FAZER 1.000 largos e'

$$2 \cdot 1000 = 2.000 \text{ ns}$$

2.I c) Quanto tempo LEVARIA UMA ADICAO EM
PEPERLINE DE 1.000 PARES DE FLUTUADORES
COM ESTAS PREMISSAS ?

DE ACORDO COM TABELA 2.3 A

CADA 9NS E ENTRABA NO LOG., DOUT E
JAHVOU FAZ A PRIMAVERA ADICAO APES
ISSO, E TEMS MILHOES NO FATOR DE 7

Logo Temos o SEGUINTE :

$$2 \cdot 999 + 3 = 2007 \text{ ns}$$

PREMISSAS EXCELENTE
RESTANTE

2.I D) O tempo necessário para buscar e armazenar pode variar consideravelmente se os operandos / os resultados são armazenados em diferentes níveis da hierarquia da memória. Suponha que a busca de um cache de Nível 1 leva 2 ns, a busca de cache nível 2 é de 5 ns.

Busca memória principal LEVA 50 ns.

O que acontece com o pipeline quando há um nível 1 cache miss é:

Mais busca de memória operando(s).

O que acontece quando há falta de nível 2?

Ocorrendo cache miss 1:

CACHE LEVEL 1: 2 ns

CACHE LEVEL 2: 7 ns

Ocorrendo cache miss 2:

CACHE LEVEL 1: 2 ns

CACHE LEVEL 2: 5 ns
MEMÓRIA PRINCIPAL: 57 ns

2.2) EXPLIQUE como UMA FILA, IMPLEMENTADA Em
HARDWARE NA CPU, Pode ser usada PARA MELHORAR O
DESEMPENHO DE um CACHE WRITE-THROUGH.

Com a FILA, como WRITE-THROUGH, ELA
ESCRVE IMEDIATAMENTE NA MEMORIA, COM
A FILA, IRÁ DIXAR AS MAIS RECENTES
NO FIMAL DA FILA, E QUE ERAM NO
INICIO DA FILA, SERIA ESCRITA NA MEMORIA
ISSO MELHORARIA A ESCRITA CONTINUANDO,
ESCRITAS REITERADAS ESTARIAM NO FIMAO DA FILA,
MELHORANDO O USO DA CACHE E PASSAMENTO.

2.3) LEMBRE-SE DO EXEMPLO ENVOLVENDO

LEITURAS DE CACHE DE UMA MATRIZ BI DEMENSIONAL (PÁGINA 22), COMO UMA MATRIZ MAIOR É UM CACHE MAIOR AFETAM O DESEMPENHO DOS DOIS PARES DE LOOPS ANINHAOS? E QUE ACONTECERIA SE $MAX = 8$ E O CACHE PODER ARMazenar 4 LINHAS? QUANTAS FALHAS OCORREM NA LEITURAS DE A NO PRIMEIRO PAR DE LOOPS ANINHAOS? QUANTAS FALHAS OCORREM NO SEGUNDO PAR?

AO TIRAR UMA MATRIZ MAIOR, MAIS DADOS PODERIAM SER CARREGADOS, POIS POSSIVELMENTE O CACHE NÃO CONSEGUERIA ARMazenAR A LINHA INTEIRA DA MATRIZ. DESSA FORMA TERÍAMOS UMA EXECUÇÃO MAIS LENTA, PAGANDO OS DOIS LOOPS.

4 ELEMENTOS DA MATRIZ, 8º PAR DE LOOPS TIRADA 76 CACHE MISSES, 2 PARA CADA LINHA DA MATRIZ
O 2º PAR DE LOOP TIRADA 64 CACHE MISSES PARES COM O AUMENTO DOS NO CACHE MISSES

2.5) A ADEGAÇÃO DE CACHE A MEMÓRIA

VIRTUAL A UM SISTEMA VON NEUMANN

ALTERA SUA DESIGNAÇÃO PARA SISTEMA

SISD? E QUANTO À ADEGAÇÃO DE

PIPELINING? VARIOS PROBLEMAS?

MULTITRADING DE HARDWARE?

* NÃO, PORQUE NA ADEGAÇÃO DA CACHE E

MEMÓRIA VIRTUAL PÔI PARA AJUSTAR O GARGALO
DA ARQUITETURA DE VON NEUMANN, PARA ASSIM

EVITAR ACESO A MEMÓRIA PRINCIPAL, PORQUE

O ACESSO A MEMÓRIA PRINCIPAL É CUSTOSO.

ISSO NÃO MUDA CARACTERÍSTICA SISD

(SINGLE INSTRUCTION STREAM, SINGLE DATA

STREAM)

* NÃO, PORQUE → PIPELINING PODE

PERMETER COMPARAÇÕES COMO HARDWARE PARALELO,

PORQUE AS UNIDADES FUNCIONAIS SÃO REPLICADAS

LOGO, A PIPELINING É TRATADO COMO

MAIS EXTENSÃO DO MODELO BÁSICO VON NEUMANN.

- * NÃO, porque o pipelining é comparado a MULTIPLE ISSUE, logo não altera SISD
- * NÃO, porque hardware multithreading apenas provê de uma thread que consiga gerir o sistema contendo executando seu trabalho. E a tarefa (task) seja entrincheda. Por exemplo, se uma tarefa (task) precise esperar dados da memória esteja pronto (suspensa). Logo o sistema com hardware multithreading carrega outra thread, porque tem switchings (trans) muito rápidos entre as threads mas isso não modifica características SISD.

2.7) DISCUTA OS DIFERENÇAS ENTRE

GPM E UM PROCESSADOR VETORIAL

Podem EXECUTAR O SEGUINTE CÓDIGO:

$$sum = 0.0$$

For ($i=0; i < n; i++$) {

$$Y[i] += a * X[i]$$

$$sum += Z[i] * Z[i]$$

3

* PROCESSADOR VETORIAL SE O TAMANHO DO

PROCESSADOR VETOR JÁ É $\frac{n}{k}$, Todo soma EXECUTADO EM UNICO CICLO, logo podemos FAZER TUDO EM PARALELO, (LOAD, ADD, STORE) CASO FOR MENOR QUE $\frac{n}{k}$, será mais LENTO PARA TERMINAR A OPERAÇÃO.

* GPM: A GPM TEM O MESMO PRINCÍPIO COMUM, TEMOS O OVERHEAD QUE O SISTEMA ~~PRECISA~~ PRECISA ENVIAR OS DADOS PARA GPM REAL TIME, PROCESSAMENTO DE MATRIZ, USO DE THREADS.

2.10) a) SUPONHA QUE UM PROGRAMA DEVE EXECUTAR 10^{12} INSTRUÇÕES PARA RESOLVER UM PROBLEMA PARTICULAR. SUPONHA AINDA QUE UM SISTEMA DE PROCESSADOR ÚNICO PODE RESOLVER O PROBLEMA EM 10^6 SEGUNDOS ($1,1,6$ diss). Portanto, EM MÉDIA, O SISTEMA DE PROCESSADOR ÚNICO EXECUTA 10^6 OU 1 MILHÃO DE INSTRUÇÕES POR SEGUNDO.

Agora, suponha que o programa PARALELO USA P PROCESSADORES, CADA PROCESSADOR IRÁ EXECUTAR $\frac{10^{12}}{P}$ INSTRUÇÕES E CADA PROCESSADOR DEVE ENVIAR 10^3 ($P-1$) MENSAGENS. FINALMENTE, SUPONHA QUE NÃO HAJA SÓBRE CARGA ADICIONAL NA EXECUÇÃO DO PROGRAMA PARALELO. O QUE SE FAZ É QUE O PROGRAMA SERÁ CONCLUIDO APÓS CADA PROCESSADOR EXECUTAR TODAS AS SUAS INSTRUÇÕES E ENVIAR TODOS AS SUAS MENSAGENS. NÃO HAVERÁ ATRASES DEVEU A COISAS COMO JITTER POR MENSAGENS.

SUPONHA QUE SEJA NECESSARIO 10³

PARA ENVIAR UMA MENSAGEM. QUANTO TEMPO O PROGRAMA LEVARÁ PARA SER EXECUTADO com 1000 PROCESSADORES, SE CADA PROCESSADOR FER TODOS RÁPIDO QUANTO O MELHOR PROCESSADOR NO MUNDO O PROGRAMA SERIA FOI EXECUADO?

cada

$\frac{10^{12}}{1000}$ É ENVIAMENTO DO PROGRAMA

TEMOS TENER 10³ PARALELO PARA

SE PLES ESTAS EN PARALELO, ENTÃO

DEMONSTRARÁ 10³ PARA QUE AS INSTRUÇÕES SEJAM EXECUTADAS, FAZ

100 10⁶ INSTRUÇÕES POR SEGUNDO.

LOGO SÓ TEMOS 10³(1000-1) MENSAGENS

O PERMITIRÁ 999 * 10⁸ MENSAGENS.

SE DEMONSTRA 10³ PARA ENVIAR

UMA MENSAGEM ENTÃO $999 * 10^8 * 10^3$

ESSO RESULTA 999.10¹¹

$$\frac{999 * 10^8 * 10^3}{999 + 10^3} = 999.10^11$$

2. 10) b) SUPONHA QUE LEVE 10^3 segundos
PARA ENVIAR UMA MENSAGEM. QUANTO TEMPO
O PROGRAMA LEVARÁ PARA SER EXECUTADO
COM 1000 PROCESSADORES?

* USANDO MESMA IDEIA

$$30 \times 50 + 10^3 = 13562,5 \text{ dia}$$

2.15) a) SUPONHA QUE UM SISTEMA DE MEMÓRIA DE MEMÓRIA COMPARTILHADA USE COERÊNCIA DE CACHE DE ESPIONAR E CACHE DE WRITE-BACK.
SUPONHA TAMBÉM QUE O NÚCLEO O TENHA A VARIÁVEL X EM SEU CACHE E EXECUTE A ATRIBUIÇÃO $X = 5$. FINALMENTE, SUPONHA QUE O NÚCLEO I NÃO TENHA X EM SEU CACHE E, APÓS A ATUALIZAÇÃO DO NÚCLEO O PARA X, O NÚCLEO I TENTA EXECUTAR $Y = X + 1$ QUAL VALOR SERÁ ATRIBUIDO A Y? Por que?

NA COERÊNCIA DE SNOOPING CACHE, TODOS OS NÚCLEOS DO SISTEMA DE MEMÓRIA COMPARTILHADA SÃO INFORMADOS QUANDO A LINHA DE CACHE, EM WRITE-BACK CACHES, OS DADOS NOSS É ESCrito. IMEDIATEMENTE NA MEMÓRIA COM. WRITE-THROUGH. ELE O DADO ATUALIZADO É MARCADO COMO SUJO, QUANDO UMA NOVA LINHA DE CACHE DO NÚCLEO A LINHA SÓ É ESCRITA NA MEMÓRIA.

O novo valor de $y=5$, já no Núcleo 2

vai ser acessado na memória, a lenha
de cache sua pelo núcleo 0 é
atualizada.

Q.15)

b) Suponha que o sistema de
memória compartilhada na parte anterior
use um protocolo baseado em diretórios.

Qual o valor será atribuído a y ?

Porque?

Quando usamos um protocolo de cache
baseada em diretórios, existe uma estrutura
de dados chamada de diretórios. Ela armazena

um TAG (status) de cada linha de cache.

Somente os núcleos usam o

variáveis em suas memórias locais

será chamada. Logo o valor y será
o que estava armazenado na memória

O Núcleo 1 não utilizou o x

2.35)

c) VOCÊ PODE SUGERIR COMO OS PROBLEMAS ENCONTRADOS NAS DUAS PRIMEIRAS PARTES PODEM SER RESOLVIDOS?

NA PRIMEIRA PARTE, O PROBLEMA É QUE TODOS OS NÚCLEOS SÃO CONNIVENTES QUANDO UMA VARIÁVEL É MODIFICADA EM UMA LINHA DE CACHE. NA SEGUNDA, OS NÚCLEOS ENVOLVEMOS SÃO CONVERGENTES, LOGO ISSO É POSSÍVEL DE CONFLITOS, PORQUE OUTRO NÚCLEO QUERIA ACESSAR ESSA VARIÁVEL, PORQUE O VALOR DA MEMÓRIA É NÃO-ATUALIZADO. ~~POSS~~

PODEMOS CRIAR UMA TABELA PARA ARMAZINHAR AS MUDANÇAS (STATMS) DAS VARIÁVEIS QUE ESTÃO ARMazenADAS NA CACHE. ~~O~~ PODEMOS CRIAR UMA VARIÁVEL, PODER NÚCLEO QUE MUDA A VARIÁVEL, PODER CONSULTAR A TABELA E SE INCLUIR SUGERIR DAI ATUALIZAR A MEMÓRIA. ASSIM, TODOS OS NÚCLEOS SÃO INFORMADOS DE MUDANÇAS NA MEMÓRIA. DESSE FORMA, RETERANDO O PROBLEMA DE EXISTÊNCIA

2. 16) SORONHA QUE O TEMPO DE EXECUÇÃO

DE UM PROGRAMA SERIAL SEJA DADO POR

$T_{\text{serial}} = n^2$, onde AS UNIDADES DE TEMPO DE EXECUÇÃO ESTÃO EM MICROSEGUNDOS.

SORONHA QUE UM PARALELIZAGÃO DESTE PROGRAMA TENHA TEMPO DE EXECUÇÃO

$$T_{\text{parallel}} = \frac{n^2}{P + \log_2 P} \quad \text{ESCREVA UM PROGRAMA}$$

QUE ENCONTRE ACELERAÇÃO E EFICIÊNCIAS DESSE PROGRAMA PARA VÁRIOS VALORES DE n E P

EXECUTE SEM PROGRAMA com $n = 10, 20, 30 \dots 320$

E $P = 1, 2, 3 \dots 128$ O QUE ACONTECE COM OS AMMENITOS DE VELOCIDADE E EFICIÊNCIAS QUANDO

P AMMENITO E n É MANTIDO FIXO?

O QUE ACONTECE QUANDO P É fixo e n É AMMENITADO?

PROGRAMA AQUI



```
c Q2.16.c •
1_prova > c Q2.16.c > ...
1  /*num computadores*/
2  float n = 10;
3
4  //processador
5  float p = 1;
6
7  /*Tempo de execução serial em
8  microsegundos*/
9  float Tserial;
10
11 /*Tempo de execução paralela em
12  microsegundos*/
13  float Tparalelo;
14
15 for(int i=0;i<6;i++){
16     for(int j=0;j<8;j++){
17         Tserial = pow(n,2);
18         Tparalelo = pow(n,2)/p + log2(p);
19         cout << "O tempo de execucao serial sera " << Tserial << "us e o tempo de execucao paralela sera " << Tparalelo
20         <<"us para n = "<<n<<" e p = "<<p<<endl;
21         p*=2;
22     }
23     p=1;
24     n*=2;
25 }
```

* QUANDO n É MANTIDO FIXO E p AUMENTA,

o tempo de execução serial é mantido constante e tempo de execução paralela diminui mais rapidamente quando o valor de n é maior que p . Quando p é mantido fixo e n aumenta, o tempo de execução cresce 4 vezes e o tempo de execução cresce proporcional a n .

2. 16) b) SO PONHA QUE $T_{PARALELO} =$

T_{SERIAL} , ~~SO~~ SO PONHA TAMBÉM QUE
P + $T_{OVERHEAD}$

CONFORMES P E AUMENTAMOS O TAMAÑO
DO PROBLEMA. MOSTRE SE $T_{OVERHEAD}$
CRESCE MAIS LENTAMENTE DO QUE T_{SERIAL} ,
A EFICIÊNCIA PARALELA AUMENTA, CONFORME
AUMENTAMOS O TAMAÑO DO PROBLEMA.

SE VERIFIQUAMOS A FUNÇÃO DA EFICIÊNCIA
NO MOMENTO EM QUE $T_{OVERHEAD}$ CRESCE
MAIS LENTAMENTE QUE T_{SERIAL} A
EFICIÊNCIA DO TEMPO PARALELO SERÁ
EMODIFICADA DEVIDO T_{SERIAL} CRESCE
NO QUANTO QUANTO MAIOR É O PROBLEMA.
ASSIM O TEMPO PARALELO É LIMITADO
PELO TEMPO SÉRIAL.

MOSTRE QUE SE, POR OUTRO LADO,
 $T_{OVERHEAD}$ CRESCE MAIS RÁPIDO DO QUE
 T_{SERIAL} , A EFICIÊNCIA DIMINUI
A MEDIDA QUE AUMENTAMOS O
TAMAÑO DO PROBLEMA?

PELA FóRMULA Temos

$$E = \frac{S}{P} = \frac{\left(\frac{T_{total}}{T_{parallel}} \right)}{P}$$

o TEMPO PARALELO SÓ IRÁ AUMENTAR
MAIS RAPIDO SE AUMENTAMOS o
TEMPO SERIAL. SE FAZEMOS $\frac{P}{=}$
A EFICIÊNCIA DO TEMPO PARALELO
DEMI UNIRÁ.

2.19) SOPONHA $T_{SERIAL} - D_n$ E $T_{PARALELO} - D_n$
 $= \frac{n + \log_2(p)}{p}$ ONDE OS TEMPOS ESTÃO
 EM MICROSEGUNDOS. SE AUMENTARMOS
 P PARA UM FATOR DE K, CONCENTRE
 NMA FORMULA DE QUANTO PRECISAMOS
 AUMENTAR N PARA MANTER A EFICIÊNCIA
 CONSTANTE. EM QUANTO DE VENDO AUMENTAR
 N SE DOBRARMOS O NÚMERO DE
 PROCESSOS DE 8 PARA 16? O PROGRAMA
 PARALELO É ESCALAVEL?

VAMOS MANEIRAR A FAMÍLIA:

$$E(k'n, kp) = E(n, p)$$

$$\frac{n}{p} = \frac{1}{\frac{n}{p} + \log_2(p)}$$

$$\frac{k'n}{kp} = \frac{\frac{1}{\frac{n}{p} + \log_2(p)}}{\frac{k'n}{kp}} + \log_2(kp)$$

$$\frac{kp}{k'p} = \left(\frac{k'n}{kp} + \log_2(kp) \right) =$$

$$\frac{P}{n} \left(\frac{n}{P} + \log_2(P) \right)$$

$$\left(I + \frac{kP}{Kn} \log_2(kP) \right) = \left(I + \frac{P}{n} \log_2(P) \right)$$

$$\left(\frac{kP}{Kn} \log_2(kP) \right) = \frac{P}{n} \log_2(P)$$

$$\frac{k}{K'} \log_2(kP) = \log_2(P)$$

$$\frac{k}{K'} \log_2(kP) = \log_2(P)$$

$$\log_2(kP) = \frac{k}{K} \log_2(P)$$

$$\log_2(k) + \log_2(P) = \frac{k}{K} \log_2(P)$$

$$k' = k \left(\frac{\log_2(k)}{\log_2(P)} + 1 \right)$$

* PARA $K=1$ E $P=8$ temos $K'=1$.

AMMENTANDO O NÚMERO DE PROCESSOS

POK UM FATOR DE 2 com $K=2$ e

$P=8$ temos $K'=2,66$ PARA UM PROGRAMA

PARALELO SER ESCALAVEL TEMOS

QUE AMMENTAR O TAMAÑO DO

PROBLEMA NA MESMA PROPORÇAO

QUE AMMENTAMOS A QANTIDADE DE
PROCESSOS. PARA $K=1$ TEMOS $K'=1$,

MAS QUE AMMENTAMOS A

QANTIDADE DE PROCESSOS, O

PROGRAMA PARALELO NÃO É ESCALAVEL.

2. 20) UM PROGRAMA QUE OBTÉM SPEEDUP LINEAR É FORTEMENTE ESCALONÁVEL?
EXPLIQUE SÓMA RESPOSTA

SIM, PORQUE MAIS PROGRAMA COM
SPEEDUP LINEAR SIGNIFICA QUE

$$T_{PARALELO} = \frac{T_{SERIAL}}{P}$$

SPEEDUP DE MAIS PROGRAMA PARALELO É
 $S = T_{SERIAL} / T_{PARALELO}$, LOGO SPEEDUP LINEAR

$$T_{PARALELO} = P \cdot T_{SERIAL}$$

DESENHO

$$E = Sp = P \cdot I$$

SERÁ SEMPRE CONSTANTE.

2.21) Bob tem um programa que desejá
cronometrar com dois conjuntos de dados,
dados de entrada 1 e dados de entrada 2,
para ter uma ideia do que esperar antes
de adicionar funções de tempo para
ao código em que está interessado, ele
executa o programa com dois conjuntos
de dados e o tempo de resposta de shell
no unix

\$ time ./bobs prob2 < input data 1

real 0m0.001s

user 0m0.001s

sys 0m0.000s

\$ time ./bobs -prob2 < input -data 2

real 2m2.239s

user 2m0.007s

sys 0m0.151s

A função de cronometro que Bob está usando
tem resolução de milissegundos. Bob deve estar
para cronometrar seu programa com o
primeiro conjunto de dados? E o segundo conjunto
de dados? Porque ou porque não?

NÃO DEVE USAR com 1º CONJUNTO DE DADOS, Porque A RESOLUÇÃO PRECISA SER MAIOR, PORQUE DESSA FORMA O TÉMPO QUE O PROGRAMA DEMORA PARA EXECUTAR. ISSO É CORRE Porque O PROGRAMA EXECUTAM MUITO RÁPIDO. PORTANTO, DEVE UTILIZAR RESOLUÇÃO MAIOR com MELHORES EXECUÇÕES EM ALGUM TIPO.

com 2º CONJUNTO DE DADOS, A RESOLUÇÃO DE MELHORES É SATISFAZENTA, Porque PERCEBEMOS O TÉMPO DE EXECUÇÃO DO PROGRAMA. ISSO OCORRE PORQUE O PROGRAMA EXECUTAM MUITO LENTO Em comparação A DADOS 1º.

2.24) SE VOCÊ NÃO FEZ ISSO NO CAPÍTULO
I, TENTE ESCRIVER PSEUDOCÓDIGO
PARA SOMA GLOBAL ESTRUTURAOS
EM ARVORE, QUE SOMA OS ELEMENTOS
DO LOC BIN-CTS. CONSIDERE PRIMEIRO
COMO ISSO PODE SER FEITO EM UMA
CONFIGURAÇÃO DE MEMÓRIA COMPARTILHADA.
EM SEGUNDA, CONSIDERE COMO ISSO
PODE SER FEITO EM CONFIGURAÇÕES
DE MEMÓRIA DISTURBADA, NA
CONFIGURAÇÃO DE MEMÓRIA COMPARTILHADA,
QUAIS VARIÁVEIS SÃO COMPARTILHADAS
E QUais São PAIVAVAS?

NA CONFIGURAÇÃO DE MEMÓRIA COMPARTILHADA
AS TAREFAS DEVE REALIZAR FAZER O LOCK
DO bin-cts ANTES QUE REALIZAR O
INCREMENTO. O ARRAY BIN-CTS E
ARRAY OF ELEMENTOS SERIA VARIÁVEL
COMPARTILHADA ENTRE AS TAREFAS.
A VARIÁVEL ARRAY LOCAL-bin-cts

SERÁ PREVADA, QUE ADMAZINA O RESULTADO.

2º ETAPA DO PROCESSO, SERÁ CHAMADA

LOCK BIN-COUNTS E SOMA OS ARRAYS

DESSA FORMA:

bin-count[i];

DATA[i];

MY-FIRST-I = ;

MY-LAST-I = ;

local-BIN-COUNT[BIN-COUNT-SUM]

FOR (i = MY-FIRST-I; I < MY-LAST-I; i++) {

INT b = FENDEN (DATA[i]);

local-BIN-COUNT[b]++;

}

LOCK (BIN-COUNT) {

} SOMA OS ARRAYS (BIN-COUNT LOCAL-OR.)

}

COM A MEMÓRIA DISTRIBUÍDA,
CADA TARDETA DEVE TER LOCAL-BIN-COUNTS
(VARIAVÉIS PROTEGIDA) E NÃO COMPARTILHARÁ A
SOMA DO BIN-COUNTS TOTAL. A ÚNICA
VARIAVÉL COMPARTILHADA NESTA SITUAÇÃO SERÁ
O ARRAY DE DATA[I]. CADA TARDETA ESTARÁ
RESOLVENDO O INTERVALO DE ELEMENTOS
E RECEBER OU ENVIAR SEU SOMATÓRIO PARA
OUTROS REALIZarem A SOMA.

O PSEUDO-CÓDIGO É:

DATA[I];

VIS D TARDETA(SIZEBINCOUNT) {

MY-FIRST-I = 0...;

MY-LAST-I = ...;

LOCAL-BIN-COUNT[SIZEBINCOUNT];

For (i = MY-FIRST-I; i < MY-LAST-I; i++) {

INT b = FIND-BIN(DATA[i]);

LOCAL-BIN-COUNT[b]++;

}

IF (DATA ENVIAR) {

} ELSE {

NOVA TABELA

SONA ARRAYS (local-BTR-comida ARRAY cima)

3 ENVIAR PARA PROXIMO MUSCLE

5.6) Escreva um programa OpenMP que determine o agendamento padrão de threads para paralelos. Sua entrada deve ser o número de iterações, e sua saída deve ser quantas iterações de um loop for paralelizado são executadas por um thread. Por exemplo se houver dois threads e quinhas iterações, a saída pode ser

Thread 0: iterations 0-1

Thread 1: iterations 2-3

Programa a seguir

ecutar Terminal Ajuda

```
c Q5.6.c x
1_prova > c Q5.6.c > main(int, char *[])
3 #include <stdlib.h>
4 #include <omp.h>
5
6 int main(int argc, char *argv[]) {
7     long thread_count;
8     int i, iteracoes;
9
10    thread_count = strtol(argv[1], NULL, 10);
11
12    printf("Insira a quantidade de iterações: ");
13    scanf("%d", &iteracoes);
14    printf("\n");
15
16    # pragma omp parallel num_threads(thread_count)
17
18        # pragma omp for
19
20            for (i = 0; i < iteracoes; ++i) {
21                printf("Thread %d - Iteração %d\n", omp_get_thread_num(), i);
22            }
23
24    return 0;
25 }
```

PROBLEMAS SAÍDA TERMINAL CONSOLE DE DEPURAÇÃO

competitive-and-distributed-programming/1_prova on ↗ master [!?]

>./Q5.6 2

Insira a quantidade de iterações: 4

Thread 0 - Iteração 0

Thread 0 - Iteração 1

Thread 1 - Iteração 2

Thread 1 - Iteração 3

competitive-and-distributed-programming/1_prova on ↗ master [!?] took 3s

>cap

5.8) CONSIDERE O LOOP

$$a[0] = a;$$

for ($i=1; i < n; i++$)

$$a[i] = a[i-1] + i;$$

HÁ claramente uma dependência circular
por loop, já que o valor de $a[i]$
não pode ser calculado sem o valor
de $a[i-1]$. Vou conseguir ver uma
maneira de ELIMINAR essa dependência
E PARALISAR o loop?

ANALISANDO o código podemos
SABER que é "a"

$$a[0] = 0$$

$$a[1] = a[0] + 1 = 0 + 1 = 1$$

$$a[2] = a[1] + 2 = 0 + 1 + 2 = 3$$

$$a[3] = a[2] + 3 = 0 + 1 + 2 + 3 = 6$$

$$a[4] = a[3] + 4 = 0 + 1 + 2 + 3 + 4 = 10$$

Logo constatamos ter
uma fórmula geral

$$T[i] = \sum_{j=0}^i j = \frac{i(i+1)}{2}$$

* substituindo pelo Fórmula Acima
sem DEPENDÊNCIA

~~for (i=0; i < n; i++) {
 a[i] = $\frac{i * (i+1)}{2};$
}~~

NESTE LOOP SEM DEPENDÊNCIA, PODEMOS

USAR PRAGMA PARALLEL, Porque o
RESULTADO NÃO É MAIS NOVAMENTE
ASSUMO PRESERVANDO A CONDIÇÃO DE CONSISTÊNCIA.

#pragmaomp parallel for num_threads(Thread-
count)
default(none) private(i) shared(a, n)

for (i=0; i < n; i++) {

 a[i] = $\frac{i * (i+1)}{2}$

}

5.3) MODELE QUE O PROGRAMA DE REGRA
TRAPEZOIDAL QUE USA UMA OPERATIVA
PARALELA PARA omp trap 3.c Para que a
paralela seja realizada por uma
CLASNA DE PLANEJAMENTO (TEMPO DE
EXECUÇÃO), EXECUTE O PROGRAMA
COM VÁRIAS ATRIBUIÇÕES Á VARIÁVEL
DE AMBIENTE OMP_SCHEDULE
E DETERMINE QUANTAS ITERAÇÕES SÃO
ATTRIBUIDAS A CADA THREAD. ISSO PODE
SER FEITO ALOCANDO UM ARRAY DE
ITERAÇÕES DE INT SE A FUNÇÃO
Trap ATTRIBUIDA omp_get_thread_num()
A ITERACOES[i] NA ÚLTIMA ITERAÇÃO
DO Loop For. QUAL É A ATRIBUIÇÃO
PREFERIDA DE ITERAÇÕES EM SEU SISTEMA?
Como AS PROGRAMAS SÃO GUARDADAS SÃO
DETERMINADAS?

O SISTEMA UTILIZADO SE NENHUMA
 CLASSE SCHEDULE FOR UTILIZADA
 PADRÃO É UMA PARTIÇÃO POR BLOCO,
 O SISTEMA OPERACIONAL ATENDE
 MENS TRABALHO A THREAD

(Threads - COUNT - I)

Executar Terminal Ajuda

```

C Q5.6.c C Q5.9.c X
1_prova > C Q5.9.c > main(int, char *[])
86
87 double Trap(double a, double b, int n, int thread_count, int *iterations) {
88   double h, approx;
89   int i;
90
91   h = (b-a)/n;
92   approx = (f(a) + f(b))/2.0;
93 # pragma omp parallel for num_threads(thread_count) \
94     reduction(+: approx) schedule(runtime)
95   for (i = 1; i <= n-1; i++) {
96     approx += f(a + i*h);
97     iterations[i-1] = omp_get_thread_num();
98   }
99   approx = h*approx;
100
101  return approx;
102 } /* Trap */

```

PROBLEMAS SAÍDA TERMINAL CONSOLE DE DEPURAÇÃO

```

Thread 0 - Iteração 0
Thread 0 - Iteração 1
competitive-and-distributed-programming/1_prova on ▶ master [!?] took 36s
>gcc -g -Wall -fopenmp -o Q5.9 Q5.9.c
competitive-and-distributed-programming/1_prova on ▶ master [!?]
>./Q5.9 4
Enter a, b, and n
1
100
10
With n = 10 trapezoids, our estimate
of the integral from 1.000000 to 100.000000 = 3.34950165000000e+05
Th 1 -> It 0;
Th 2 -> It 1;
Th 2 -> It 2;
Th 1 -> It 3;
Th 1 -> It 4;
Th 2 -> It 5;
Th 1 -> It 6;
Th 2 -> It 7;
Th 2 -> It 8;
Th 0 -> It 9;
competitive-and-distributed-programming/1_prova on ▶ master [!?] took 10s
>or

```

QUANDO USES A CLASNA SCHEDULE
(COMP-SCHEDULE), TEMOS 3 SEGUNDES
COMPONENTES ~~n~~
~~THREAD - COUNT~~

INTERAÇÕES SÃO ATENDIDAS, POR

EXEMPLO $n = 10$, LOGO TIMES

$$\frac{10}{4} = 2,5$$

SUCESSIVOS blocos DE INTERAÇÃO com
METADE DO TAMANHO DO DADO ANTERIOR.

5. I2) BAIXE O ARQUIVO FONTE DA
MÍTICAS E EXECUTE O COMPILADOR DO LISP.
Encontre um programa que FAÇA O PÁRCEL
DE CACHE (POR EXEMPLO, VALGRIND [4.3]) E
COMPILE O PROGRAMA DE ACORDO COM AS
INSTRUÇÕES COMPLEJAS (COM O USO DO GCC -f -O2).
NA DOCUMENTAÇÃO DO CRIADOR DE PÁRCEL
DE CACHE (POR EXEMPLO, COM VALGRIND SOBRE
VAI QUERER UMA TABELA DE SÍMBOLOS
E OTIMIZAÇÕES COMPLETA (COM USO DO GCC,
GCC -f -O2...). AGORA EXECUTE O
PROGRAMA DE ACORDO COM AS INSTRUÇÕES
NA DOCUMENTAÇÃO DO CRIADOR
DE PÁRCEL DE CACHE, USANDO A
ENTRADA $K * (K * 10^6)$, $(K * 10^3) * (K * 10^3)$
 $2 * (K * 10^6) * K$. ESCOLHA K TÃO GRANDE
QUE O NÚMERO DE FALHAS DE CACHE
DE NÍVEL 2 SEJA DA ORDEM DE 10^6
PARA PECAS MENOS UM DOS CONJUNTOS
DE DADOS DE ENTRADA.

5.I2

a) QUANTAS FALHAS DE GRAVAÇÃO DE CACHE
DE NIVEL I OCORREM COM CADA
UMA DAS TRÊS ENTRADAS?

A orden das matrizes são $8 \times 3.000.000$,

$$8.000 \times 8.000 + 3.000.000 \times 8$$

os resultados obtidos foram:

Vejamos cada uma a seguir:

MATRIZ	L2 READ	L2 READ	L2 WRITE	L2 WRITE
a) $8 \times 3.000.000$	16.000.000	36.000.000	16.000	570
b) 8.000×8.000	16.000.000	3.000.000	1.000	2.000
c) $3.000.000 \times 8$	8.000.000	3.000.000	3.000.000	3.000.000

5.I2 |

c) ONDE OCORRE A MAIORIA DOS ERROS DE GRAVAÇÃO? PARA QUais DADOS DE ENTRADA O PROGRAMA TEM MAIS ERROS DE GRAVAÇÃO?
VOCÊ PODE EXPLICAR POR QUE?

O maior número de erros de escrita ocorre com uma matriz de $8.000.000 \times 8$, NESTE SISTEMA o array Y tem orden de $8.000.000$. Isto quer dizer que nos sistemas com orden F de 8.000 a 8.000 existem 1000 entradas.

ENTRE AS VARIÁVEIS A, X E Y E

VARIAVEL QUE É ISOLADA PELA COLUNA.

5.12) \neq ONDE O CORRE A MAIORIA DOS ERROS DE LEITURA?

PARA QUais DADOS DE ENTRADA O PROGRAMA TEM MAIS ERROS DE LEITURA? VOCÊ PODE EXPLICAR POR QUÉ?

A MAIOR QUANTIDADE DE ERROS ESTA NA MATRIZ $3 \times 8.000.000$. Porque CADA ELEMENTO DA MATRIZ Δ VAI SER LIDO UMA MILHA VEZ COM AS 3 ENTRADAS. E CADA ELEMENTO TEM 64.000.000 ELEMENTOS. AO ATUALIZAR (update) $Y[i][j] += ...$ SÓ ESTÁ EXATAMENTE 64.000.000 SOMAS E OS INTEGRAIS SÃO LIMITADOS. $Y[i][j]$ É INICIALMENTE ZERO. PROBABILMENTE SE ESTÁ NA CACHE LOGO É POSSÍVEL QUE NÃO TENHA ERROS NA LISTA.

MAS POSSIVEL EXPLICAR ISSO

que ao ler x[8] põem canas

FALHAS DE CACHE E PODER QUER

NAS $3 \times 8.000.000$ QUE Têm MAIS FLAMIGOS

DO que os outros que é 8,2800

5.12) g) EXECUTE o PROGRAMA com

CADA UNA DAS TRÊS ENTRADAS, MAS SEM

USAR A CRIADOR DE PAREL DE CACHE.

Com qual ENTRADA o PROGRAMA É MAIS RAPIDO?

com qual ENTRADA o PROGRAMA É MAIS LENTO?

SMAS observações SOBRE FALHAS DE CACHE

PODEM AJUDAR A EXPLICAR AS DIFERENÇAS?

Como?

Thread 0	$3 \times 8.000.000$	0,93 s
----------	----------------------	--------

Thread 1	3.000×8.000	0,36 s
----------	----------------------	--------

Thread 2	$3.000.000 \times 3$	0,42 s
----------	----------------------	--------

O PROGRAMA MAIS LENTO É O COM
MATEZ DE ORDEN $8 \times 8.000.000$ E
MAIS RÁPIDO É 8.000×8.000 .

POSSUI FLENTES PONTO NA LEITURA
TEMPO MAIS FLNETOS E TAMBÉM
É MAIS CURTO COMPLETAMENTE.

QUANDO UM PROGRAMA SOLICITA
DADOS PARA EXECUTAR, ELÉ PODE SOLTAR
FAZER RETENTATIVAS OU ESPERA PELOS DADOS
AGORA DADOS SOLTADOS PARA ESCRITA (WRITE)
PODEM SER COLOCADOS NA FILA E NÃO
PODERÁ O PROGRAMA ASSIM, MAS FICANDO
OSINT.

O PROGRAMA COM UMA MATEZ
 $8.000 \times 8.000 \times 3$ TEM MAIS ERROS NA
ESCRITA DO QM O COM 8.000×8.000
A QUANTIDADE DE ERROS DE LEITURA
DE L2 É IGUAL

1A PARA OS 6 MESES O CUSTO
COMPUTACIONAL, COMPARANDO COM 12
DE ERROS DE LEITURA. LOCAL
O PROGRAMA 3.000 X 8.000 E'
NESS PERÍODO.

5.13) Lembrar-se do exemplo de multiplicação do vetor de matriz com a entrada 8.000×8.000 . Suponha que Thread 0 e o Thread 2 sejam atribuídos a processadores diferentes. Se uma linha de cache contém 64 bytes em 8 double, é possível que ocorra um falso compartilhamento entre os threads 0 e 2 para qualquer parte do vetor γ ?

Por que? Se a Thread 0 é a Thread 3 forem atribuídos a processadores diferentes; é possível que ocorra um falso compartilhamento entre elas para qualquer parte de γ .

Com 8.000 elementos γ está dividido como segue:

$$0 : \gamma[0], \dots, \gamma[399]$$

$$1 : \gamma[4000], \dots, \gamma[7999]$$

$$2 : \gamma[8000], \dots, \gamma[11999]$$

$$3 : \gamma[12000], \dots, \gamma[15999]$$

FALSO - COMPARTILHAMENTO
PARA OCORRER ENTRE AS THREADS 0 e 2, DEVE HAVER
ELEMENTOS DE Y QUE PERTENCE A MESMAS
LINHAS DE CACHE, MAS ASSOCIADOS A
THREADS DIFERENTES. NA THREAD 0, A LINHA
DE CACHE QUE ESTÁ MUITO "PROXIMA" DOS
ELEMENTOS ASSOCIADOS A THREADS 2. É
A LINHA QUE CONTÉM Y[1385]. MAS
MESMO QUE ESTE SEJA O PRIMEIRO ELEMENTO
DA ~~FEITA~~ LINHA DE CACHE.

LOGO → INDICE MUITO ALTO PARA O
ELEMENTO QUE PERTENCE A FEITA LINHA É
2006:

Y[1382] Y[2000] Y[2002] Y[2003] Y[2004]
Y[2004] Y[2005] Y[2006]

JÁ QUE O MESMO ÍNDICE DE UM ELEMENTO
É ASSOCIADO A 2 DE 9000, NÓS A
POSSIBILIDADE DA LINHA DA CACHE
~~PODE~~ TER ELEMENTOS QUE PERTENCE
AO SEGMENTO 0, QUANDO AO SEGMENTO
2. O RACIOCÍNIO SEMELHANTE SE
APLICA AOS SEGMENTOS 0 e 3.

5. 14) Lembre-se do exemplo do

VETOR DE MATRIZ com uma MATRIZ

$8 \times 8,000,000$, suponha que double 8

use 8 bytes de memória e que uma
LINHA DE CACHE TENHA 64 BYTES.

Suponha também que nosso sistema
conste em dois processadores DUAL-CORE.

a) Qual é o número mínimo de
LINHAS DE CACHE NECESSÁRIAS
PARA ARMazenar o VETOR Y?

Ao rodar o programa, percebemos que

a localização de $Y[0]$ na primeira
LINHA DE CACHE CONTENDO TODO
ON PARTE DE Y, Vemos que Y PODE
SER DESTACIBILDO ENTRE AS LINHAS
DE CACHE DE 8 FORMAS DIFERENTES.

Se $Y[0]$ é o PRIMEIRO ELEMENTO
DA LINHA DE CACHE,

ENTÃO A DISTRIBUIÇÃO DE X NA CACHE
SERÁ:

1º LINHA: $y[0] \dots y[7]$

SE $y[0]$ É SEGUNDO ELEMENTO
DA LINHA LOGO SERÁ

2º LINHA $y[0] \dots y[5]$

2º LINHA $y[7] \dots$

SE $y[0]$ É ULTIMO ELEMENTO DA LINHA
ENTÃO:

1º LINHA: $\dots y[7]$

2º LINHA: $y[2] \dots y[7] \dots$

COM ISSO PODEMOS PESQUISAR A

QUESTÃO A: PNE SIM PODERÁS

ARMARIZAR NA CACHE EM 1 LINHA

5. I 9)

b) QUAL É O NÚMERO MÁXIMO DE LINHAS DE CACHE NECESSÁRIO PARA ARMazenAR O VETOR Y2 NO MÁXIMO ELE OCUPARÁ DAS LINHAS DE CACHE

S. I 9)

c) SE OS LIMITES DAS LINHAS DE CACHE SEMPRE COINCIDEM COM OS LIMITES DE DOUBLE DE BYTES DE DIVERSAS MANEIRAS DIFERENTES OS COMPONENTES DE Y PODEM SER ATRIBUIDOS AS LINHAS DE CACHE?

OS LIMITES COINCIDEM,
LOGO HÁVIA APENAS 2 POSSIBILIDADES

S.I.9)

d) SE CONSEDERARMOS API NAS QMAIS

PARES DE THREADS COMPARTILHAM UM

PROCESSORES EM NOSSO COMPUTADOR?

AQUI ESTAMOS ASSUMINDO QUE OS

NÚCLEOS NO MESMO CACHE DE

COMPARTILHAMENTO DO PROCESSADOR.

PODEMOS ESCOLHER 2 DAS THREADS E

ATRIBUI-LAS A UM DOS THREADS

NOMEADOS: 0 e 1, 0, 2, 0 e 3 ou 0 e 4.

ENTÃO EXISTEM 4 ATRIBUIÇÕES POSSÍVEIS NAS

THREADS PARA OS DIRETÓRIOS.

S. 14) E) EXISTE UMA ATIVIDADES
DE COMPONENTES PARA LINHAS DE
CACHE E THREADS PARA PROCESSADORES
QUE RESULTARÁ EM NENHUM FALSO
COMPARTILHAMENTO EM NOSSO EXEMPLO?
EM OUTRAS PALAVRAS É POSSÍVEL
QUE AS THREADS ATIVADAS A UM
PROCESSADOR TENHA SEUS COMPONENTES
DE Y EM UMA LINHA DE CACHE E AS
THREADS ATIVADAS A OUTRO PROCESSADOR
TENHAM SEUS COMPONENTES EM
UMA LINHA DE CACHE DIFERENTE?

SIM, POIS QUE AS THREADS 0, E I
COMPARTILHAM UM PROCESSADOR E AS
THREADS 2 E 3 COMPARTILHAM OUTRO.
ENTÃO SE Y[0], Y[1], Y[2] E Y[3]
ENTÃO EM UMA LINHA DE CACHE Y[4],
Y[5], ..., Y[7] ESTÃO EM OUTRA.
A ESCRETA PELA THREAD 0 ONZ
NOS ENFORCAR NOS CASOS

2 e 3 THREADS, SÓ GERA MESMO

RACIOCÍNIO NOS ÍCAR FINAIS DA
A CACHE 0, e 1.

5.19) f) QUANTAS ATRIBUIÇÕES DE
COMPONENTES PARA LINHAS DE CACHE E
THREADS PARA PROCESSADORES EXISTEM?

PARA CADA MMA DAS 9 ATRIBUIÇÕES
DE THREADS PARA INDICAR, EXISTEM 8
ATRIBUIÇÕES POSSÍVEIS DE X PARA
LINHAS DE CACHE. PORTANTO TEMOS
UM TOTAL DE $9 * 8 = 32$ ATRIBUIÇÕES.

5.19) g) DESSAS ATRIBUIÇÕES, QUANTAS
RESULTADOS EM NENHUM FATO
COMPARTILHAMENTO? TEM APENAS
UMA POSSIBILIDADE, AQUELA QUE
É APRESENTADA NA LIGA, ~~EXCEÇÃO~~

QUESTÕES EXTRAS

PROGRAMMING ASSIGNMENTS

S. I) MSE OPENMP PARA IMPLEMENTAR O

PROGRAMA DE HISTOGRAMA PARALELO DESCRIÇÃO

NO CAPÍTULO 2

PROGRAMA DE

Ejecutar Terminal Ajuda

Q2.16.c Q5.1.c x

```

1_prova > c Q5.1.c > ...
78     data = malloc(data_count*sizeof(float));
79
80     /* Generate the data */
81     Gen_data(min_meas, max_meas, data, data_count);
82
83     /* Create bins for storing counts */
84     Gen_bins(min_meas, max_meas, bin_maxes, bin_counts, bin_count);
85
86     /* Count number of values in each bin */
87     #pragma omp parallel for num_threads(thread_count) default(none) \
88         shared(data_count, data, bin_maxes, bin_count, min_meas, bin_counts) \
89         private(bin, i)
90     for (i = 0; i < data_count; i++) {
91         bin = Which_bin(data[i], bin_maxes, bin_count, min_meas);
92         #pragma omp critical
93         bin_counts[bin]++;
94     }
95
96 # ifdef DEBUG
97     printf("bin_counts = ");
98     for (i = 0; i < bin_count; i++)
99         printf("%d ", bin_counts[i]);
100    printf("\n");
101 # endif
102
103    /* Print the histogram */
104    Print_histo(bin_maxes, bin_counts, bin_count, min_meas);
105
106    free(data);
107    free(bin_maxes);
108
```

PROBLEMAS SAIDA TERMINAL CONSOLE DE DEPURAÇÃO

competitive-and-distributed-programming/1_prova on ✘ master [?]

```

>./Q5.1 5 0 10 100 4
0.000-2.000: XXXXXXXXXXXXXXXX
2.000-4.000: XXXXXXXXXXXXXXXX
4.000-6.000: XXXXXXXXXXXXXXXX
6.000-8.000: XXXXXXXXXXXXXXXXXXXX
8.000-10.000: XXXXXXXXXXXXXXXXXXXX

```

competitive-and-distributed-programming/1_prova on ✘ master [?]

5.2) SUPONHA QUE JOGUEMOS DARDOS ALEATORIAMENTE EM UM ALVO DE DARDOS QUADRADO CUJO ALVO ESTÁ NA ORIGEM E OS SEUS LADOS TÊM 60 cm de COMPRIMENTO. SUPONHA TAMBÉM QUE HAJA UM CÍRCULO INSCRITO NO ALVO DE DARDOS QUADRADO.

O raio do círculo É DE 1 PÉ E Sua ÁREA É DE TRÊS². SE O PONTO que SÃO ACERTADOS PELOS DARDOS ESTÃO UNIFORMEMENTE DISTRIBUÍDOS (E SEMPRE ACERTAMOS O QUADRADO) ENTÃO O NÚMERO DE DARDOS QUE ACERTAM DENTRO DO CÍRCULO DEVE SATISFAZER APROXIMADAMENTE A EQUAÇÃO.

$$\frac{\text{Número de dardos dentro do círculo}}{\text{Total número de dardos}} = \frac{\pi}{4}$$

JÁ QUE A RAZÃO ENTRE A ÁREA DO CÍRCULO E A ÁREA DO QUADRADO É $\pi/4$, PODERMOS USAR ESTA FÓRMULA PARA ESTIMAR O VALOR DE π COM UM GRANDE NÚMERO DE NÚMEROS ALEATÓRIOS:

NUMBER IN CIRCLE = ?

for (toss = 0; toss < number of tosses; toss++) {

X = random double between -1 and 1;

Y = random double between -1 and 1;

distance squared = X * X + Y * Y;

} if (distance squared <= 1) number in circle ++;

pi estimate = 4 * number in circle / ((double) number of
tosses);

Esse é CHAMADO DE MÉTODO "MONTE CARLO",

Pois usa ALGA TÉCNICA DE OS LANÇAMENTOS DE DADOS.

ESCREVA UM PROGRAMA EM PEP8 QUE USE UM
MÉTODO DE MONTE CARLO PARA ESTIMAR PI.

LEIA O NÚMERO TOTAL DE JOGADAS ANTES DE
BITURCAR QUALQUER LINHA. USE UMA CLASSE
DE REDAÇÃO PARA EXCORTAR O NÚMERO TOTAL
DE DARDOS ACERTANDO DENTRO DO CÍRCULO E
O NÚMERO DE JOGADAS, UMA VETOR QUE AMBOS
PODEM TER QUE SER MUITO GRANDE PARA
OBTER UMA ESTIMAÇÃO Satisfatória DE PI.

CÓDIGO A SEGUIR

5.2

```
c Q5.2.c ●
l_prova > c Q5.2.c > ...
1
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <math.h>
5 #include <omp.h>
6
7 void Usage(char* prog_name);
8
9 double num_aleatorio() {
10     double numero = (double) random() / (double) RAND_MAX;
11     if((double) random() / (double) RAND_MAX < 0.5) {
12         numero *= -1;
13     }
14     return numero;
15 }
16
17 int main(int argc, char* argv[]) {
18     long int number_tosses, number_in_circle;
19     int thread_count, i;
20     double x, y, distancia;
21
22     if (argc != 3) Usage(argv[0]);
23     thread_count = strtol(argv[1], NULL, 10);
24     number_tosses = strtoll(argv[2], NULL, 10);
25     if (thread_count < 1 || number_tosses < 1) Usage(argv[0]);
26
27     number_in_circle = 0;
28     srand(a).
```

PROBLEMAS SAÍDA TERMINAL CONSOLE DE DEPURAÇÃO

```
>./Q5.2 6 20
Estimacao de pi = 3.200000000000000
competitive-and-distributed-programming/1_prova on ↗ master [?]
>./Q5.2 6 100
Estimacao de pi = 3.160000000000000
competitive-and-distributed-programming/1_prova on ↗ master [?]
>./Q5.2 6 1000
Estimacao de pi = 3.160000000000000
competitive-and-distributed-programming/1_prova on ↗ master [?]
>./Q5.2 6 10000
Estimacao de pi = 3.178400000000000
competitive-and-distributed-programming/1_prova on ↗ master [?]
>]
```

5.3) CLASSE FILHA PESO CONTA EMA F UN
ALGORITMO DE CLASSIFICAÇÃO SERIAL SIMPLES
QUE PODE SER IMPLEMENTADO DA SEGUNTE
MANEIRA:

```
Void count sort (int a [3], int n) {  
    int i, j, count;  
    int * temp = malloc (n * sizeof(int));  
    for (i = 0; i < n; i++) {  
        count = 0;  
        for (j = 0; j < n; j++)  
            if (a[j] < a[i])  
                count++;  
            else if (a[j] == a[i] && j > i)  
                count++;  
        temp [count] = a[i];  
    }  
    memcpy (a, temp, n * sizeof(int));  
    free (temp);  
}
```

A IDÉIA BÁSICA É QUE PARA CADA ELEMENTO $a[i]$ NA LISTA A , CONTAMOS O NÚMERO DE ELEMENTOS NA LISTA A QUE SÃO MENORES QUE $a[i]$. Em SEGUNDA INSERIMOS $a[i]$ EM UMA LISTA TEMPORÁRIA USANDO O SUBSCRITO DETERMINADO PELA CONTAGEM. HÁ UM PEQUENO PROBLEMA COM ESSA ABORDAGEM QUANDO A LISTA CONTÉM ELEMENTOS FÓRMAIS, UNA VZ QUE ELES PODEM SER ATRIBUÍDOS AO MESMO SLOT NA LISTA TEMPORÁRIA. O CÓDIGO LIDA COM ESSO INCREMENTANDO A CONTAGEM DE ELEMENTOS IGUAIS COM BASE NOS SUBSCRITOS.

SE $a[i] == a[j] & j < i$, ENTÓIS CONTAMOS $a[j]$ COMO SENDO "MENOR QUE" $a[i]$.
APOIS A EXECUÇÃO DO ALGORITMO, NOS SOBRECREVEMOS O ARRAY ORIGINAL PELO ARRAY TEMPORÁRIO USANDO A FUNÇÃO MEMO DA BIBLIOTECA DE STRINGS

5.3 a) SE TENTARMOS PARALELIZAR

o loop for i (o loop externo) QUAI'S

VARIÁVEIS DEVER SÃO PRIVADAS E

QUAI'S DEVER SER COMPARTILHADA?

AS VARIÁVEIS DEVER SER COMPARTILHADAS

JÁ QUE SERIA NECESSÁRIO ACESSAR A
JANELA ATÉ TER ACESSO AO SEM TANTO.

A VARIÁVEL TAMBÉM SERIA

COMPARTILHADA, MAS VIZ QUE CADA

PASSADA DO LOOP SO SERIA ESCUTADA

POR UM NÚMERO THREADS. AS VARIÁVEIS

i, j, constaram AS VARIÁVEIS INICIAIS

5.3 b) SE PARALELIZERMOS o loop for

i USANDO O ESCOPO - ESPECIFICO NA

PARTE ANTERIOR, HÁ ALGUMAS OPEN DENOMINADAS

PELÔS loop? EXPLIQUE SUA

RESPOSTA,

~~NÃO~~, NÃO HÁ NENHUMA DEPENDÊNCIA ENTRE AS ITERAÇÕES UMA VETORE QUE NÃO SERIA NECESSÁRIO TER ACESSO A UMA INFORMAÇÃO DE UMA OUTRA ITERAÇÃO DURANTE A EXECUÇÃO DO CÓDIGO. ALÉM DISSO, A VARIÁVEL TIRP PODERIA SER ESCrita PELA THREAD SEM GRANDE PROBLEMA DE CONCORRÊNCIA.

5.3) c) PODEMOS PARALELIZAR A CHAMADA PARA MEMCPY? PODEMOS MODIFICAR O CÓDIGO PARA QUE ESTA PARTE DA FUNÇÃO SEJA PARALELIZADA?

SIM, UMA CHAMADA AO MEMORY TAMBÉM PODERIA SER PARALELIZADA. A ÚNICA PECULIARIDADE SERIA EM RELAÇÃO AO TAMAÑHO NO QUAL NÃO DEVERIA SER O MESMO. ALÉM DISSO, O PRIMEIRO E O SEGUNDO ARGUMENTO DA FUNÇÃO MEMORY DEVERIA SER PONTENTOS PARA ELEMENTOS

DO ACASO A PARTIR DO ANALOGO

COMISSAO A ESCRITA - SIGME & EXEMPLO DO

CODEGO:

#pragma omp parallel num-threads(THREADS-cont.)
shared (tmp, a, n) {

int TAMNTO = n/thread - count;

int INDEX-INICIAL = omp_get_rn-
Thread() * TAMNTO;

memcpy (da [INDEX-INICIAL], dst [in-
DEX-INICIAL], TAMNTO * (SIZEOF (int)));

}

FREE (tmp);

5.3) d) ESCREVA um PÁGRAMA C QUE
ENCIMA MAIS IMPLEMENTAÇÕES PARALELAS
DA CLASSE CACOS COUNT.

CODE DO ALGORÍMO

```
ar Terminal Ajuda
c Q5.2.c u ● c Q5.3-d.c x
1_prova > c Q5.3-d.c > ...
22     printf("%d ", a[i]);
23 }
24     printf("\n");
25 }
26
27 int main(int argc, char* argv[]) {
28     long int number_tosses, number_in_circle;
29     int thread_count, i, j, n, count;
30     srand(0);
31     thread_count = strtol(argv[1], NULL, 10);
32     n = strtol(argv[2], NULL, 10);
33     int * a = malloc(n* sizeof(int));
34     geraMatriz(a, n);
35     //imprimeMatriz(a, n);
36     int * temp = malloc(n* sizeof(int));
37     double start = omp_get_wtime();
38 #pragma omp parallel for num_threads(thread_count) \
39     default(private) private(i, j, count) shared(a, n, temp, thread_count)
40     for (i = 0; i < n; i++) {
41         count = 0;
42         for (j = 0; j < n; j++)
43             if (a[j] < a[i])
44                 count++;
45             else if (a[j] == a[i] && j < i)
46                 count++;
47             temp[count] = a[i];
48     }
49     memcpy ( a , temp, n * sizeof(int));
50     double finish = omp_get_wtime();
51     free(temp );
52     printf("Tempo estimado %e segundos\n", finish - start);
53     //imprimeMatriz(a, n);
54     return 0;
55 } /* main */
```

5.3) e) como o DE SEMPRENAO DE SMA

PARALELIZAÇÃO da classificação por contagem
SE compara com a classificação por contagem
SINAL? como ELAS SE COMPARA à JUNÇÃO
DA BIBLIOTICA SIRIA + SORT?

$\theta(n \log n)$

$\theta(n^2)$ e COUNT-SORT é $\theta(n^2)$

P/H	5000	10000
1	130,62 <u>(3,59)</u>	725,81 <u>(3,73)</u>
2	164,62	262,12
3		

* Qsort

5.4) LEMBRE-SE DE QUE, QUANDO RESOLVEMOS
UM GRANDE SISTEMA LINEAR, GERALMENTE
USAMOS A ELIMINAÇÃO DE GAUSSIANA SEGUINDO
PELA SUBSTITUIÇÃO REVERSA. A ELIMINAÇÃO
GAUSSIANA CONVERTE UM SISTEMA LINEAR
 $n \times n$ EM UM SISTEMA LINEAR TRIANGULAR
SUPERIOR USANDO AS "OPERAÇÕES DE LINHA"
... NO LIVRO RESO.

- a) DETERMINAR SE O LOOP EXTERNO DO
ALGORITMO orientado por LINHA PODE SER
PARALELIZADO.
- O Loop mais externo não é possível
de ser PARALELIZADO uma vez que
EXISTE DEPENDÊNCIA ENTRE AS ITERAÇÕES.
- O SISTEMA É RESOLVIDO DE BAIXO PARA
CIMA (DA ULTIMA LINHA ATÉ A PRIMEIRA)
E OS RESULTADOS DAS LINHAS MAIS
ACIMA.

5.4 b) DETERMINE SE O Loop

INTERNO DO ALGORITMO SISTENADO
POR LINHA PODE SER PARALELIZADO.

O loop mais interno é capaz de ser
PARALELIZADO, NO ENTANTO, EXISTE uma
REGIÃO CRÍTICA A REALIZAR O

" $\times [LINHA] =$ " fique FAZENDO COM QUE
APENAS uma Thread REALIZE A SUBSTÂNCIA
POR VEZ. (SERIAL)

5.4) c) DETERMINE SE O (SEGUNDO) Loop

EXTENSO do algoritmo orientado a coluna
Pode ser PARALELIZADO.

O loop mais EXTERNO NÃO É POSSÍVEL
DE SER PARALELIZADO POR QUE EXISTE
DEPENDÊNCIA ENTRE AS ITERAÇÕES, PORQUE
OS LOOPS SIGUINTE AO SER CRIADO

A SUBTRAÇÃO REALIZADA NO loop
ANTERIOR PÔDE PODER PROCEDE

5.4 d) DETERMINAR SE O LOOP INTERNO

DO ALGORITMO ORIENTADO A COLUNA PODE
SER PARALELIZADO.

O LOOP MAIS INTERNO É POSSÍVEL

POIS NÃO EXISTE REGIÃO CRITICA
COMO NO "ROW-ORIENTED".

5.4 e) ESCREVA UM PROGRAMA OPENMP PARA

CADA UM DOS LOOP QUE VOCÊ DETERMINOU

QUE PODERIA SER PARALELIZADO. VOCÊ PODE

ACHAR A ÚNICA DIRETIVA ÚTIL - QUANDO

UM BLOCO DE CÓDIGO ESTÁ SENDO

EXECUTADO EM PARALELO E UM SUB-BLOCO

DEVE SER EXECUTADO POR APENAS UMA THREAD,

O SUB-BLOCO PODE SER MODIFICADO POR
UMA DISSETA UNICA # PRAGMA OMP. AS
THREADS NA EQUIPE DE EXECUÇÃO SERÃO

BLOQUEADOS NO FINAL DA DIRETIVA ATÉ

QUE TODOS OS THREADS TENHAM
CONCLUIDO.

PROGRAMA ORIENTADO POR
LINHA ABAIIXO!

```
#pragma omp parallel for num_threads(thread_count) default(none) \
private(col) shared(x, b, a, n, row)
for (col = row + 1; col < n ; col++) {
    double valor = a[row*n + col]*x[col];
    #pragma omp critical
    x[row] -= valor;
}

x[row] /= a[row*n + row];
}
```

PROGRAMA Orientado por
COLUNA ABAIIXO!

```
#pragma omp parallel for num_threads(thread_count) default(none) \
private(row) shared(x, b, n)
for (row = 0; row < n; row++) {
    x[row] = b[row];
}

for (col = n - 1; col >= 0; col--) {
    x[col] /= a[col*n + col];

    #pragma omp parallel for num_threads(thread_count) default(none) \
private(row) shared(x, b, a, n, col)
    for (row = 0; row < col; row++) {
        x[row] -= a[row*n + col]*x[col];
    }
}
```

5.4) F) MODELO QUE SEN LOOP PARALELO

com uma classificação de PLANEJAMENTO (TEMPO DE EXECUÇÃO) E TESTE o PROGRAMA com VARIOS PLANEJAMENTO. SE o SISTEMA TRIANGULAR SUPERIOR TEM 10.000 VARIANTEIS QUAL CRONOGRAMA OFERECE o MELHOR DESEMPENHO?

PADRAO	ESTATICO I	ESTATICO II	
6835	6751	6538	
DINAMICO I	DINAMICO II	DINAMICO III	CRITICO
6840	7273	6750	6725

OBSERVANDO os resultados o melhor TVE com EXECUCAO PADRAO. Possui a EXECUCAO PADRAO. GANTADO - TVE RECENTES PIGEM. A PIGEM EXECUCAO É DINAMICA PORQUE ELA ADICIONA A ONTARIO NO SISTEMA.