"ANALISE E IMPLEMENTAÇÃO DA FFT NO HIPERCUBO"

① HISTÓRICO

"PERFORMANCE ANALYSIS OF THE FFT ALGORITHM ON A SHARED MEMORY PARALLEL ARCHITECTURE"

"BEST AND WORST MAPPINGS FOR THE Ω NETWORK"

ZARKA CVETANOVIC - IBM J.R.D - VOL 31 - July 1987

- PROPOSIÇÃO: ESTUDO DE UMA REDE OMEGA COM PIPELINE: IBM-RP3 NYU ULTRACOMPUTER

- MODELO DETERMINÍSTICO

- TRANSFORMADA DE FOURIER:

a) Se a 1ª iteração é sem conflito, então todas as iterações seguintes são sem conflito.

b) Se a 1ª iteração tem conflito MAXIMO, então todas as iterações seguintes tem conflito MAXIMO.

c) Numero de mapeamentos sem conflito: $2^{N/2 \log N} = \sqrt{N^N}$

- BIT REVERSAL & SHUFFLE-EXCHANGE

a) Em um mapeamento identidade, estas operações não podem ser realizadas sem que haja conflito. Mas a propriedade de máx conflito não se preserva e existem configurações que realizam estas operações sem conflito.

b) Stone, H.S "High Performance Computer Architecture", pp 312:

"Cvetanovic [1986] mostrou que as duas operações são incompatíveis com o shuffle (rede OMEGA), na medida em que, se os dados são armazenados de modo que as operações de butterfly ocorrem sem conflito então as operações de "bit-reversal" resultam em máximo conflito na rede"

Cvetanovic nunca disse isto. No artigo em questão, a única afirmação que fiz é a seguinte (pp 440).

"De modo a simplificar a análise, nós desconsideraremos o custo associado a permutações de "bit-reversal" para todos os algoritmos analizados neste artigo"

- EM ABERTO:

. ANÁLISE PARA O CASO EM QUE A COMUNICAÇÃO E O PROCESSAMENTO SE SOBREPÕEM.

. IDENTIFICAÇÃO DE QUAIS MAPEAMENTOS RESULTAM EM MÁXIMO OU MÍNIMO NÚMERO DE CONFLITOS.

② ANÁLISE DETERMINÍSTICA DA FFT NO HIPERCUBO

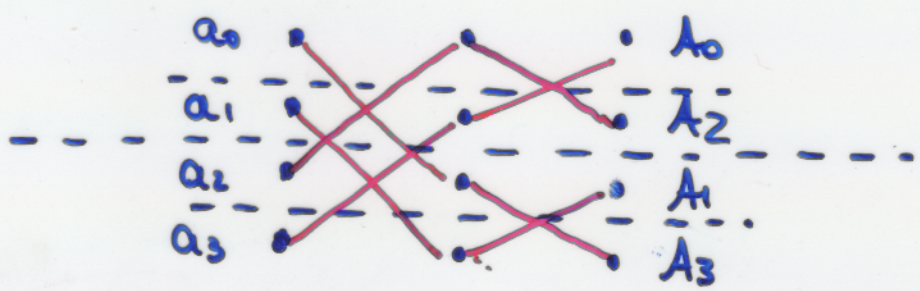
. OBTER QUAL A ALOCAÇÃO INICIAL DOS DADOS QUE RESULTA EM:

(i) MAPEAMENTO ÓTIMO (= mínimo conflito)

(ii) MELHOR R/C.

(i)

MAPEAMENTO ÓTIMO É AQUELE EM QUE A COMUNICAÇÃO OCORRE SOMENTE ENTRE VIZINHOS E APENAS EM LOG N PASSOS. (N = Nº DE PROCESSADORES EXISTENTES).



N PONTOS

N = 1, 2 ou 4 proc.

- EXISTEM $C^{M}_{M/N}$ COMBINAÇÕES POSSÍVEIS PARA A ALOCAÇÃO DOS DADOS, OU, NO CASO EM QUE M=N, N! COMBINAÇÕES. O PROBLEMA É DETERMINAR QUANTAS E QUAIS SÃO AS COMBINAÇÕES ÓTIMAS.

- A DETERMINAÇÃO DESTAS CONFIGURAÇÕES É IMPORTANTE POIS PERMITE QUE OUTROS TIPOS DE OPERAÇÃO SEJAM REALIZADAS SOBRE OS MESMOS DADOS COM O MENOR Nº DE CONFLITOS POSSÍVEL.

- PARA N=4, AS SEGUINTES CONFIGURAÇÕES SÃO ÓTIMAS:

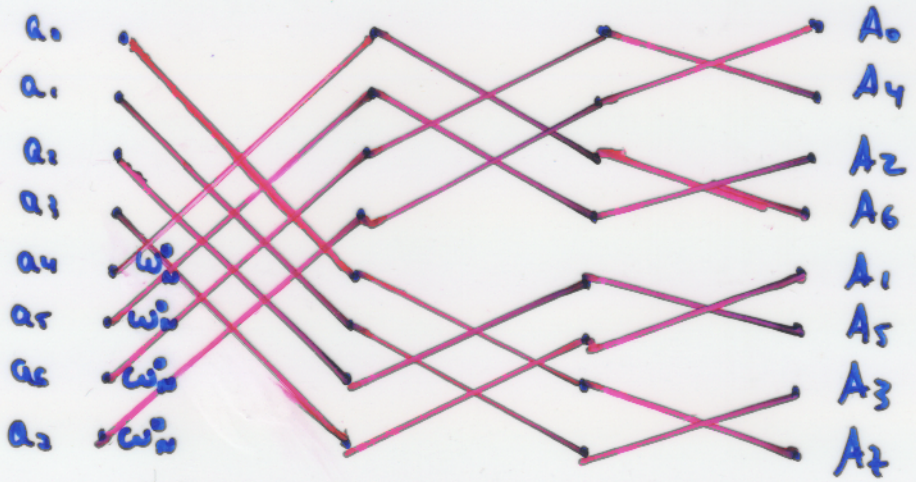
0	1	2	3
1	0	3	2
2	3	0	1
3	2	1	0

0	1	2	3
2	3	0	1
1	0	3	2
3	2	1	0

- ESTE ESTUDO AINDA ESTÁ PARA SER CONCLUÍDO.

(ii)

- O MELHOR R/C É AQUELE QUE RESULTA EM UM MENOR TEMPO DE EXECUÇÃO DO PROGRAMA.



- A QUESTÃO É: DADO QUE TEMOS N PROCESSADORES E M PONTOS, QUANTOS PONTOS ALOCAR PARA CADA PROCESSADOR? $\frac{M}{N}$? $\frac{2m}{N}$?

- PREMISSAS:

  - R & C se sobrepõem
  - M pontos, $M = 2^m$
  - N proc, $N = 2^n$
  - m, n inteiros
  - $t_c > t_p$

- TEMPO TOTAL DE EXECUÇÃO PARA UMA DISTR. HOMOGÊNEA, PODE SER EXPRESSO POR:

$$T_t = \frac{M}{N}\left(\log\left(\frac{M}{N}\right)t_p + \log N \cdot t_c\right)$$

$$T_t = \frac{M}{N}\left((m-n)t_p + m\,t_c\right)$$

- Se fizermos $n' = n-1$, ou seja, dobrarmos o nº de pontos por processador, teremos

$$T_t' = \frac{2M}{N}\left[(m-(n-1))t_p + (n-1)t_c\right)$$

Se NÃO houve ganho, a seguinte desigualdade deve ser verdadeira!

$$(m-n)t_p + mt_c < 2\left[(m-(m-1))t_p + (m-1)t_c\right]$$

$$\boxed{\frac{t_p}{t_c} > \frac{1}{2+m-n}}$$

- DEPENDENDO DOS VALORES DE TP E TC O USO DE UM CUBO MENOR PODERÁ SER VANTAJOSO OU NÃS. CURIOSAMENTE, SE $M=N$, temos

$$\boxed{\frac{t_p}{t_c} > \frac{1}{2}}$$

(3) IMPLEMENTAÇÃO NO T800

OPÇÕES REALIZADAS:

1. Mapeamento identidade:

   É ótimo. Produz melhor balanceamento de carga.

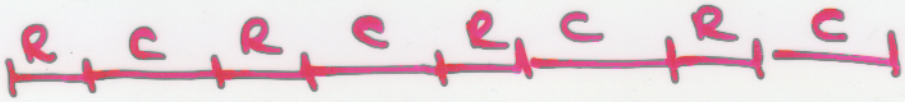
2. Cálculo dos pesos W_N^c recursivamente:

   gasta menos memória e comunicação do que ter uma tabela pré-calculada em todos os nós.

3. O levantamento de t_p e t_c no modelo de melhor R/C só é válido para N=4. Acima deste valor deve-se incluir latência adicional para acesso aos canais adicionais, que estarão mapeados na memória externa.

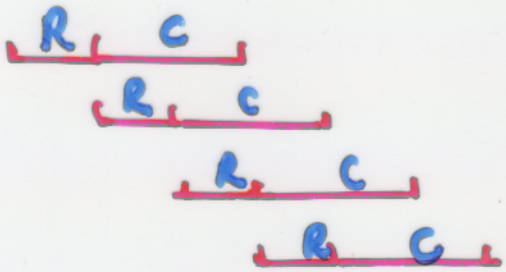
4. Comunicação // processamento através do uso de "pipeline".

   (i) SEM PIPE (R & C serial)



$$t = 4(R+C)$$

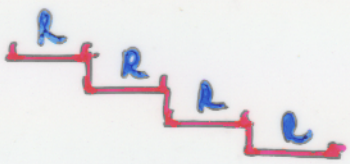
(i) Pipe ideal (R||R e C||C)



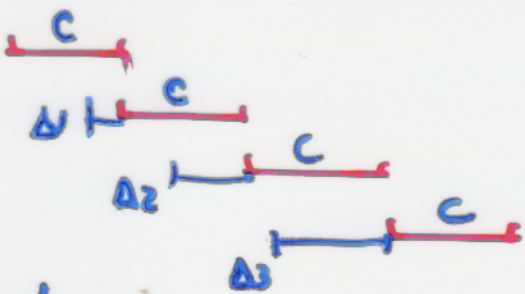
$t = 4R + C$

(ii) Pipe Real (só R em II)

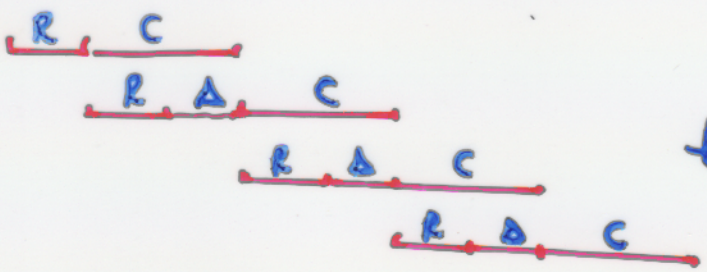
(a) Buffer = N



$t = 4C + R$



(b) Buffer = 1



$t = 4C + R$

VANTAGEM: GANHO DE MEMÓRIA.