VETORIZAÇÃO DE MALHAS PARA SUPERCOMPUTADORES

Li Kuan Ching

Orientador: Siang Wun Song

Computador de alto desempenho é uma necessidade que cresce dia a dia em áreas como aerodinâmica, engenharia genética, inteligência artificial, exploração de petróleo, previsão de tempo, dentre outras aplicações científicas e na engenharia. Sem estas máquinas de grande poder computacional, muitos dos desafios para o avanço da ciência não podem ser enfrentados dentro de um prazo razoável.

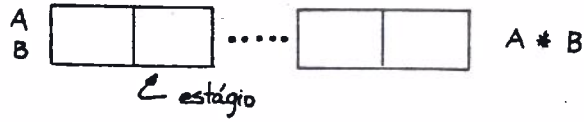
Supercomputador, uma das classes destas máquinas de alto desempenho, é caracterizado por possuir um processador vetorial e pelo seu alto poder computacional. Citamos, como exemplo, algumas destas máquinas disponíveis hoje no mercado: Cray 2 e Cray Y-MP da Cray Research, linha SX da NEC Corporation, Convex da Convex Computer Corporation, etc.

Como é feita uma operação aritmética num supercomputador?

Sejam A e B dois números dados e C, o resultado da operação aritmética aplicado ao A e B. O resultado C será obtido somente se A e B tenham completado todos os estágios definidos para a operação. Por exemplo:

A, B: números

C: resultado da operação onde $C := A * B$



Sejam agora A e B dois vetores de tamanho N dados e C o vetor resultado obtido da operação aritmética aplicado ao A e B.

O processo para obtermos o vetor resultado C é ilustrado na figura abaixo. Podemos ter como analogia à idéia acima a linha de montagem automobilística, onde em cada

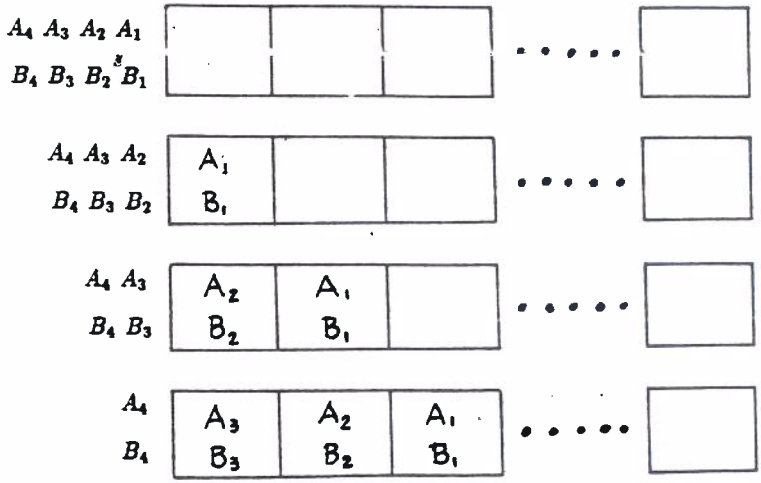
estágio da linha é feito um tipo específico de serviço. Observe que existem instruções vetoriais nessas máquinas. Por exemplo:

A, B: vetores de tamanho 100

C: vetor resultado, onde $C(I) := A(I) * B(I)$

Uma instrução vetorial:

$C(1:100) := A(1:100) * B(1:100)$



Mostraremos, no que se segue, um dos aspectos dos compiladores que detectam o uso de vetores para que gere com sucesso os códigos vetorizados, mediante teste da Dependência de dados.

A detecção da existência de relações de dependência de dados é importante no sentido de determinar quando duas operações ou comandos podem ser executados em paralelo. Por exemplo, na seguinte seqüência de instruções

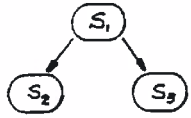
S_1 : $A := B + C$

S_2 : $D := A + 2$

S_3 : $E := A * 3$.

S_1 e S_2 não podem ser executados em paralelo, pois S_2 depende do valor de A que é computado em S_1. Assim, chamaremos de Dependência Verdadeira ou "flow dependence",

desde que os dados fluem de S_1 a S_2 e é denotado por $S_1\ \delta\ S_2$. O mesmo ocorre com S_3 ($S_3\ \delta\ S_1$). Portanto, S_1 deve ser executado antes de S_2 e S_3. Representando em grafos de dependência de dados, com as flechas indicando as relações, temos:



$S_1\ \delta\ S_2$

$S_1\ \delta\ S_3$

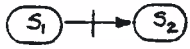
Note-se que S_2 e S_3 não possuem ligações entre si e podem, portanto, ser executados em paralelo.

Há ainda dois outros tipos de dependência de dados que são importantes. Na seqüência de instruções

S_1 : $A := B + C$

S_2 : $B := D/2$

S_1 usa o valor de B antes que S_2 dê um novo valor ao B. Se S_1 usa o valor "antigo" de B, este deve ser executado antes de S_2. Essa dependência é chamada de anti-dependência. Denotemos por $S_1\ \bar{\delta}\ S_2$ e a sua representação em grafos de dependência de dados:



$S_1\ \bar{\delta}\ S_2$

O terceiro tipo de dependência é mostrado na seguinte seqüência de instruções:

S_1 : $A := B + C$

S_2 : $D := A + 2$

S_3 : $A := E + F$

S_3 dá um novo valor a A após S_1 já tê-lo dado. Se S_1 é executado depois de S_3, A conterá um valor diferente. Portanto S_1 deve preceder S_3. Chamamos então de dependência de saída ("output dependence") e denotamos por $S_1\ \delta^o\ S_3$.

Devemos notar que dependência de dados depende também do controle de fluxo. Por exemplo:

S_1 : $A := B + C$

if $(X >= 0)$ then

S_2 : $A := 0$

else

S_3 : $D := A$

endif

Observa-se que as relações $S_1\ \delta^o\ S_2$ e $S_1\ \delta\ S_3$ são dependências verdadeiras, mas não $S_2\ \delta\ S_3$, mesmo que em S_2 dê-se um novo valor a A e em S_3 usa-se o valor de A, e que S_3 venha depois de S_2 na seqüência de instruções. Como S_2 e S_3 são de "ramos" diferentes do mesmo comando **if**, o valor de A usado em S_3 nunca viria de S_2.

Para falarmos um pouco mais sobre dependência de dados, mostraremos então a dependência de dados em "malhas" ("Loops"). A noção inicial da execução de uma seqüência de instruções em "malha" pode ser dada em:

do $I = 1, 3$

S_1: $A(I) := B(I)$

do $J = 1, 2$

S_2: $C(I, J) := A(I) + B(J)$

end do

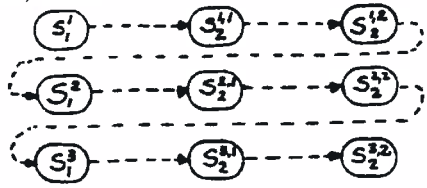
end do

Observa-se que:

S_1 é executado 3 vezes: S_1^1, S_1^2, S_1^3;

S_2 é executado 6 vezes: $S_2^{1,1}$, $S_2^{1,2}$, $S_2^{2,1}$, $S_2^{2,2}$, $S_2^{3,1}$, $S_2^{3,2}$.

Para ilustrar o caso, tem-se:



As linhas tracejadas indicam a ordem da execução da "malha".

Para acharmos dependência de dados em "malhas", são examinados os vetores e seus índices. Na seguinte "malha":

$$\begin{aligned} &\text{do } I = 2, N \\ S_1: &\quad A(I) = B(I) + C(I) \\ S_2: &\quad D(I) = A(I) \\ &\text{end do} \end{aligned}$$

É verdadeira a relação $S_1\ \delta\ S_2$, pois na iteração i, o valor de $A(i)$ é calculado em S_1^i e será usado pelo S_2^i na mesma iteração da "malha". Visto que a dependência se mantém na mesma iteração da "malha", dizemos que $S_1\ \delta_{=} S_2$.

Na seguinte "malha", similar ao anterior,

$$\begin{aligned} &\text{do } I = 2, N \\ S_1: &\quad A(I) = B(I) + C(I) \\ S_2: &\quad D(I) = A(I-1) \\ &\text{end do} \end{aligned}$$

Continua verdadeira $S_1\ \delta\ S_2$, mas na iteração i, S_2^i usará valores de A que foram obtidos da iteração anterior da "malha" pelo S_1^{i-1}. Desde que a dependência resulte da iteração $i-1$ para iteração i, dizemos que $S_1\ \delta_{<}\ S_2$ (a relação é $\delta_{<}$ pois $i-1 < i$).

Um terceiro exemplo similar é dado abaixo:

$$\begin{aligned} &\text{do } I = 2, N \\ S_1: &\quad A(I) = B(I) + C(I) \\ S_2: &\quad D(I) = A(I+1) \\ &\text{end do} \end{aligned}$$

Nesta "malha", para qualquer iteração i, S_2^i usará elementos de A na qual lhe será atribuído um novo valor em S_1^{i+1}. Visto que S_2 deverá usar um valor "antigo" de A, a relação de antidependência $S_2\ \overline{\delta}\ S_1$ é verdadeira. Como esta relação resulta da iteração i para a iteração $i+1$, temos que $S_2\ \overline{\delta}_{<} S_1$.

O uso de $=$ ou $<$ como índice de δ é chamado de Direção da dependência de dados, pois este lhe indica a direção da relação de dependência na "malha". Em "malhas" encaixadas, há uma direção para cada "malha", por exemplo:

```
      do I = 1,N
        do J = 2,N
S1:       A(I,J) := A(I,J-1) + B(I,J)
S2:       C(I,J) := A(I,J) + D(I+1,J)
S3:       D(I,J) := 0.1
        end do
      end do
```

Temos as seguintes relações de dependência:

$$S_1 \quad \delta_{=,<} \quad S_1$$

$$S_1 \quad \delta_{=,=} \quad S_2$$

$$S_2 \quad \overline{\delta}_{<,=} \quad S_3$$

Após termos visto a noção da dependência de dados, veremos agora então a vetorização de "malhas".

"Malhas" podem ser vetorizadas se examinando o grafo da dependência de dados, for constatada a ausência de ciclos no grafo da dependência. Por exemplo, na seguinte "malha":

```
      do I = 1,N
S1:     A(I) := B(I)
S2:     C(I) := A(I) + B(I)
S3:     E(I) := C(I+1)
      end do
```

tem o seguinte grafo da dependência de dados:

$\longrightarrow$ dependência verdadeira

$\longrightarrow$ anti-dependência

Por não haver ciclos no grafo da dependência, este pode ser vetorizado completamente, mesmo que sejam necessárias algumas alterações na seqüência de instruções da "malha". Como $S_3\ \overline{\delta}\ S_2$, S_3 deve preceder S_2 na "malha" vetorizada:

$$S_1: \ A(1:N) := B(1:N)$$

$$S_3: \ E(1:N) := C(2:N+1)$$

$$S_2: \ C(1:N) := A(1:N) + B(1:N)$$

O exemplo a seguir será dado com ciclo no grafo da dependência:

do $I = 2, N$

S_1: $A(I) := B(I)$

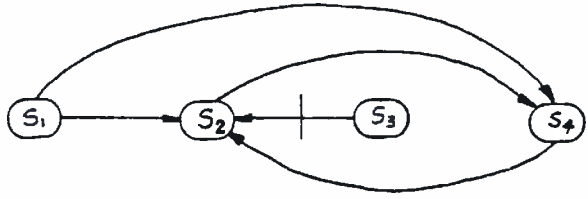
S_2: $C(I) := A(I) + B(I-1)$

S_3: $E(I) := C(I+1)$

S_4: $B(I) := C(I) + 2$

end do

e o seu grafo da dependência de dados:



Observe que as instruções S_2 e S_4 formam um ciclo no grafo da dependência de dados. Todos os comandos do ciclo devem ser executados numa "malha". No entanto,

outras instruções podem ainda ser vetorizadas. O exemplo dado fica:

S_1: $A(2:N) := B(2:N)$

S_3: $E(2:N) := C(3:N+1)$

do $I = 2, N$

S_2: $C(I) := A(I) + B(I-1)$

S_4: $B(J) := C(I) + 2$

end do

Para finalizar este texto, queremos enfatizar que a concepção da vetorização de "malhas" já está incorporada em modernos compiladores dos supercomputadores e um problema importante que incide sobre o texto é a detecção de paralelismo e geração de código para execução em processadores paralelos.

Bibliografia

Padua, D.A. & Wolfe, M.J., *Advanced Compiler Optimizations for Supercomputers*, Communications of ACM, December, 1986, pp. 1184-1201.