CAP-387(2016) – Tópicos Especiais em Computação Aplicada: Construção de Aplicações Massivamente Paralelas

Aula 42: Coarray Fortran

Celso L. Mendes, Stephan Stephany

LAC / INPE

Emails: celso.mendes@inpe.br, stephan.stephany@inpe.br

Coarray Fortran - CAF

- Histórico
  - Introduzido em 1998, como extensão a Fortran95
  - Muitos anos de experiência, principalmente em Crays
  - Incluído no padrão Fortran2008
- Objetivos
  - Extender Fortran minimamente para suportar paralelismo
  - Expressar a extensão de forma natural e intuitiva para os programadores de Fortran
  - Manter nível de desempenho no código compilado, mesmo sem troca de mensagens explícita

Coarray Fortran - CAF

- Especificação:
  - R.W.Numrich & J.K.Reid, “Co-Array Fortran for Parallel Programming”, ACM Fortran Forum, 17(2):1-31, 1998
- Implementações
  - Disponível em vários compiladores Fortran atuais, comerciais ou abertos
    - Cray, Intel, PGI, GNU, Open64, etc
  - Tipicamente, traduzido para Fortran normal, com chamadas a uma biblioteca de comunicação
    - Ex: Fortran+MPI, Fortran+GasNET, etc.

CAF – Aspectos Gerais

- Modelo de execução
  - Programa composto por réplicas que executam simultaneamente, de forma assíncrona
  - Cada replica é chamada de “imagem” (*image*)
  - Número total de imagens: função *num_images()*
  - Índice da imagem do processo local: *this_image()*
  - Índices das imagens variam de *1* a *num_images()*
- Extensão de Fortran adotada
  - Arrays com uma co-dimensão adicional, separada, para indicar a distribuição do array pelas imagens

CAF – "Hello-World"

```
program hello
      integer :: my_id, num_procs
      my_id   = this_image()
      num_procs = num_images()
      sync all
      if (my_id .EQ. 1) print *,"Numero de Imagens: ",num_procs
      sync all
      print *,"Imagem ",my_id
end program hello
```

CAF – "Hello-World" (cont.)

- Compilação num Cray-XE6:
  - ftn –h caf –o prog prog.f90
- Execução em 8 processadores: 2 nós, 4 procs/nó:

> aprun -n 8 -N 4 prog
Numero de Imagens: 8
Imagem 1
Imagem 2
Imagem 3
Imagem 4
Imagem 5
Imagem 6
Imagem 7
Imagem 8

CAF - Coarrays

- Declaração de coarrays
  - Sintaxe adicional em Fortran:

    real, dimension(100), codimension[*] :: x

    real :: x(100)[*]

    Declara um array de 100 elementos em cada imagem
  - Qualquer parte de *x* pode ser acessada remotamente pelas várias imagens
  - Variáveis escalares também podem ser coarrays
  - Coarrays têm o mesmo tamanho em todas as imagens
  - Coarrays podem ser alocados (*ALLOCATE*)

CAF – Acesso a Coarrays

- Exemplo:

  integer :: a(10)[*], b(10)[*] ← **Declara coarrays *a,b***

  b(:) = a(:)[n] ← **Copia *a[n]* para todos os *b's***

  - Supondo *n=3*:
    - Elementos *a(1:10)* da imagem *3* são copiados para os arrays *b* em todas as imagens
    - Na prática, há um broadcast de *a(:)[n]*
    - Caso em que *n > num_images()* é um erro
  - Em geral: *()*=acesso local *[]*=acesso remoto

CAF – Acesso a Coarrays

- Outro exemplo:

```
real :: x[*]
if (this_image() == 1) then
      read *,x
      do k=2,num_images()
            x[k] = x
      end do
end if
sync all
```

Declara coarray *x* (escalar)

Apenas imagem 1 lê o valor *x*

Copia valor lido para imagem *k*

Aguarda por término das cópias

- Observações:
  - Não é permitido usar notação vetorial na codimensão:
    x[2:num_images()]=x é inválido!
  - Não há ordem implícita sobre o término das várias cópias; por exemplo, a cópia para a imagem 5 poderia terminar *antes* da cópia para a imagem 4!

CAF - Sincronização

- Sincronização implícita no início do programa
  - Garante que os coarrays existem, em todas as imagens
- Comandos implícitos de sincronização em Fortran
  - Barreira: *sync all*
    → sincroniza todas as imagens
  - Barreira parcial: *sync all (lista(:))*
    → sincroniza apenas as imagens na lista
  - Uso recomendável:
    - sincronizar antes de acessar coarrays
      → garante que acessos remotos não ocorram antes do uso
    - sincronizar depois de acessar coarrays
      → garante que o novo valor é disponível para todas as imagens

CAF - Exemplo

```
program reduce
      integer :: my_rank, num_procs, i, result
      integer     :: coarray[*]
      my_rank   = this_image()
      num_procs = num_images()
      if ( (my_rank/2)*2 .EQ. my_rank ) then
            coarray[my_rank] = my_rank;
      else
            coarray[my_rank] = my_rank * (-1);
      end if
      result = 0
      sync all
      if (my_rank .EQ. 1) then
            do i=1, num_procs
                  result = result + coarray[i]
            end do
      end if
      sync all
      if (my_rank .EQ. 1) print *,"CAF_size: ",num_procs,"  result: ",result
      sync all
end program reduce
```

Inicialização do coarray:

- *coarray[i]=i* se *i* for par
- *coarray[i]=-i* se *i* for ímpar

→ *{-1,2,-3,4,-5,6,...,NUM_IMAGES()}*

- Soma: *NUM_IMAGES/2*

Proc. 1 realiza as somas, serialmente, usando dados de cada um dos outros processadores (ou seja, comunicação unilateral!)

CAF – Exemplo (cont.)

- Compilação num Cray-XE6:
  - ftn –h caf –o prog_caf prog_caf.f90
- Execução em 1024 processadores: 32 nós, 32 processadores/nó:

  > aprun -n 1024 -N 32 prog_caf

  CAF_size: 1024 result: 512

CAF – Outros Detalhes

- Codimensão pode ser composta: *real :: z(10)[5,*]*
  - Valor de "*" é feito de modo a preencher *num_images()*
    - Se *num_images*=20: codimensões=[5,4]
    - Se *num_images*=60: [5,12]
    - Se *num_images*=18: [5,4] , mas [4,4] e [5,4] não existem
  - Elementos *z(:)[1,4]* estão na imagem 16

(percorrimento das codimensões na ordem de Fortran)

[1,1]	[1,2]	[1,3]	[1,4]
[2,1]	[2,2]	[2,3]	[2,4]
[3,1]	[3,2]	[3,3]	[3,4]
[4,1]	[4,2]	[4,3]	[4,4]
[5,1]	[5,2]	[5,3]	[5,4]

  - Se uma referência é maior que *num_images()*: ERRO!

CAF – Outros Detalhes

- Restrições de I/O para as imagens:
  - *stdin* válido apenas na imagem-1
  - *stdout* e *stderr* válidos em todas as imagens
- Coarrays podem ser usados em I/O
- Coarrays não são interoperáveis com C!

CAF – Outro Exemplo

- Busca do valor máximo:

```
real :: a(10)
real :: maximum[*]
call random_number(a)
maximum = maxval(a)
sync al
if (this_image() == 1) then
        do image = 2, num_images()
                maximum = max(maximum, maximum[image])
        end do
        do image = 2, num_images()
                maximum[image] = maximum
        end do
end if
sync al
```

Toda a busca feita pela Imagem-1

Imagem-1 encontra o máximo

Imagem-1 distribui o máximo