OpenMP

Programowanie współbieżne... (8)

Andrzej Baran 2010/11

LINK: http://kft.umcs.lublin.pl/baran/prir/index.html

Literatura (podstawa opracowania):

Hermanns, M. (2002). Parallel Programming in Fortran 95 using OpenMP. [pdf] http://link.aip.org/link/?CPHYE2/8/117/1

Treść

- Wstęp
- Dyrektywy
  - Klauzule, warunki
- Funkcje i procedury czasu wykonania
- Zmienne środowiskowe

Klauzule

- Klauzule (warunki) rozszerzają dyrektywy i precyzują, ustalają sposób wykonywania się obszarów równoległych programu pracujacego w modelu OpenMP
- Wzorzec (fixed fortran)

  C$OMP *directive-name optional_clauses...*

  !$OMP *directive-name optional_clauses...*

  *$OMP *directive-name optional_clauses...*
- Wzorzec (free fortran)

  !$OMP directive-name optional_clauses...

Zakresy danych

- Klauzule zakresów danych określają jak poszczególne zmienne są traktowane i kto (jaki wątek) widzi ich zawartość oraz kto może je zmieniać
- W przykładach będziemy najczęściej używać pary dyrektyw !$omp parallel / !$omp end parallel ale inne dyrektywy są również możliwe (patrz odpowiednia Tabela)

Atrybuty zakresów danych

- private(lista)
- shared(lista)
- none
- default(private | shared | none)
- firstprivate(lista)
- lastprivate(lista)
- copyin(lista)
- copyprivate(lista)
- reduction(operator:lista)

private(lista)

!$omp parallel private(a, b)

Zmienne a, b będą miały różne wartości w każdym wątku – będą to zmienne prywatne wątku. Każdy wątek rezerwuje pamięć na te zmienne – wzrasta zajętość pamięci. Przed i po obszarze *private* wartości zmiennych a, b nie są zdefiniowane.

shared(lista)

```
!$omp parallel shared(c, d)
```

Obie zmienne c, d są widoczne przez każdy wątek, są globalne (dzielone) dla wątków z obszaru objętego dyrektywą parallel/end parallel. Wątki mogą je zmieniać dowolnie. Nie zwiększa się zajętości pamięci.

Użycie dyrektywy $omp flush wymusza uaktualnienie zmiennych wspólnych.

Ponieważ każdy wątek może zapisywać zmienne dzielone, więc wynik zapisu nie jest określony (*racing condition*).

default(private|shared|none)

default(private | shared | none)

Klauzula ta upraszcza procedurę w przypadku gdy większość zmiennych w obszarze działanie dyrektywy !$omp... posiada charakter private|shared.

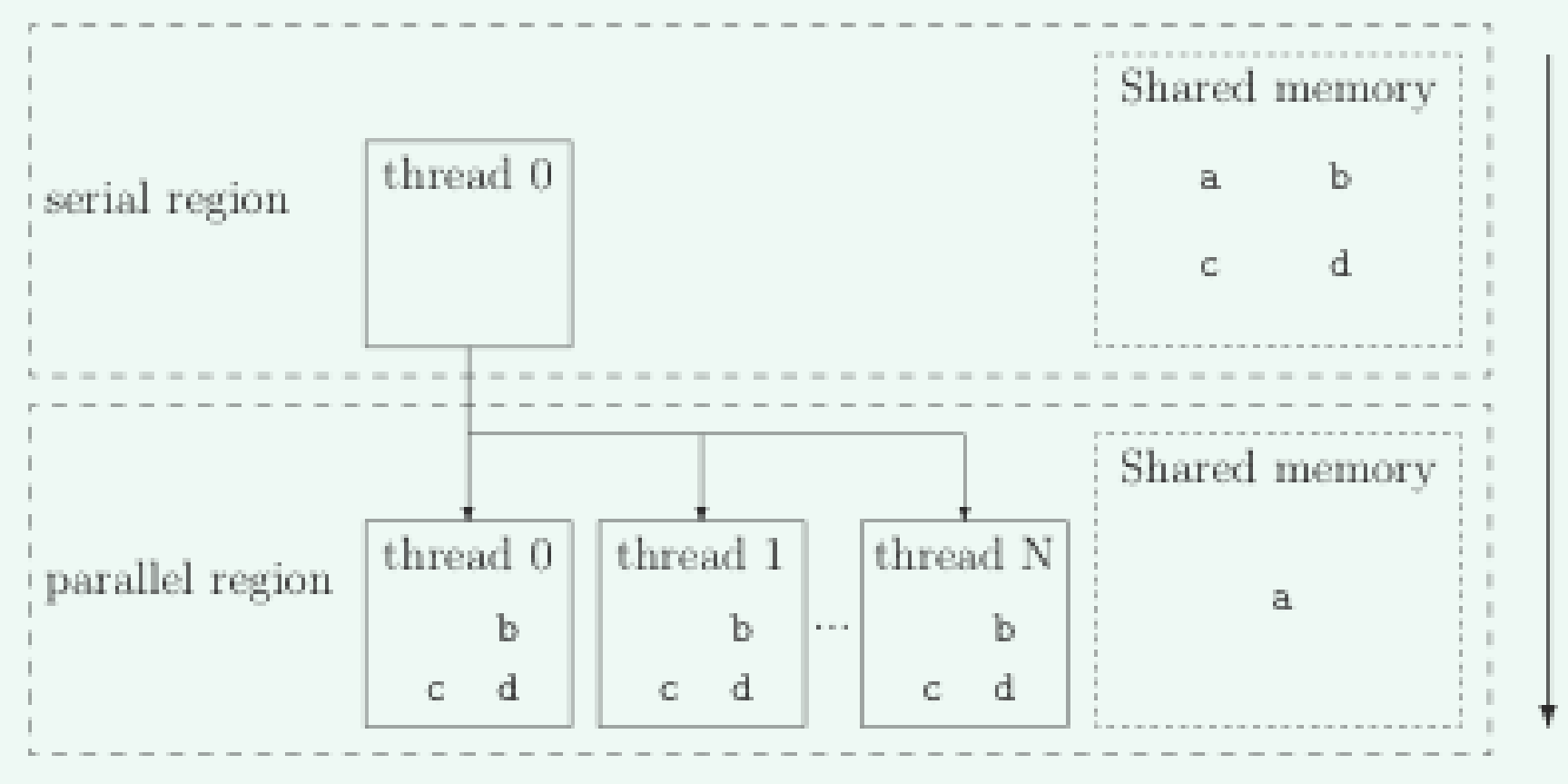
!$omp parallel default(private) shared(a)

oznacza, że wszystkie zmienne z wyjątkiem a są prywatne.

default(none) oznacza, że zakres każdej zmiennej w obszarze dyrektywy musi być jawnie określony. Wyjątek: zmienna jest threadprivate lub jest indeksem pętli.

default(private|shared|none)

!$omp parallel default(private) shared(a)

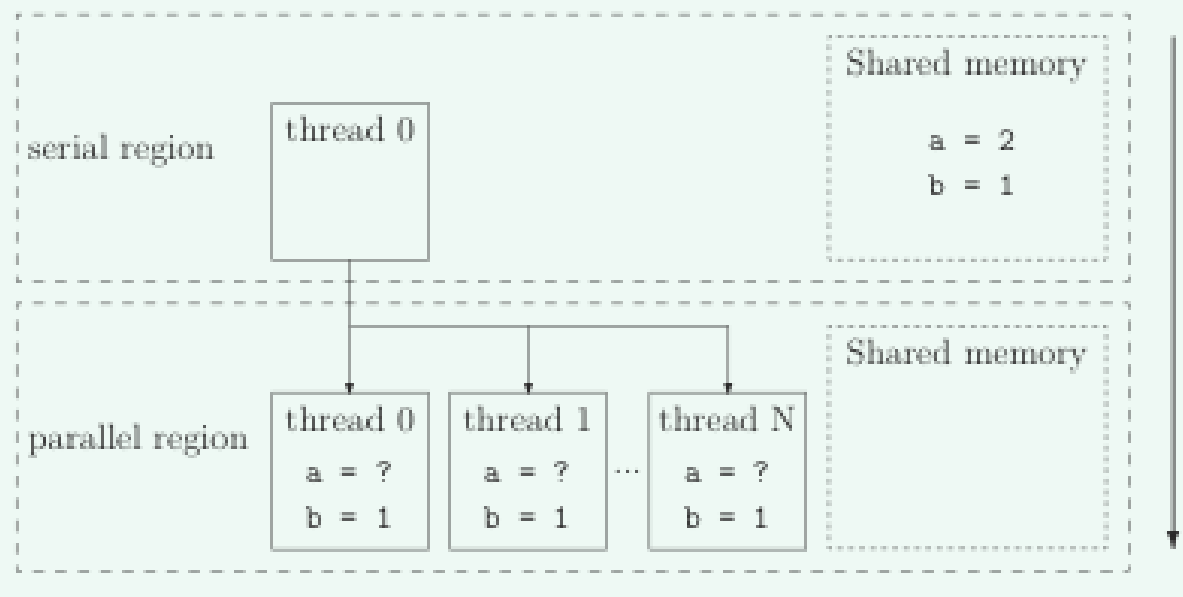


firstprivate(lista)

a = 2; b = 1

!$omp parallel private(a) firstprivate(b)

Zmienna b ma w obszarze działania dyrektywy taką samą wartość jak w obszarze sekwencyjnym (zmienna a ma wartość nieokreśloną)



lastprivate(lista)

Przykład 1

```
!$OMP DO PRIVATE(i) &
!$OMP   LASTPRIVATE(a)
    do i = 1, 1000
          a=i
    end do
!$OMP END DO
```

Zmienna LASTPRIVATE(a) po wyjściu z obszaru !$OMP ma wartość określoną, równą w tym wypadku 1000.

Przykład 2

```
!$OMP SECTIONS LASTPRIVATE(a)
!$OMP SECTION
    a=1
!$OMP SECTION
    a=2
!$OMP END SECTIONS
```

Wątek leksykalnie ostatni (wykonujący drugą sekcję) podstawi pod a wartość 2 po zakończeniu sekcji. Jeśli a nie występuje w ostatniej sekcji, to będzie nieokreślone.

copyin(lista)

Przykład

```
!$OMP THREADPRIVATE(a)
!$OMP PARALLEL
    a = OMP_get_thread_num()
!$OMP END PARALLEL

!$OMP PARALLEL
    …
!$OMP END PARALLEL

!$OMP PARALLEL COPYIN(a)
    …
!$OMP END PARALLEL
```

Używając warunku COPYIN(a) zmiennej a określonej jako THREADPRIVATE można nadać jej wartość równą tej jaką posiada ona w wątku głównym (*master*).

W przykładzie obok, zmienna a ma w drugim obszarze wartość taką, jak w pierwszym, ale w trzecim obszarze równoległym jej wartość jest zerem, bo *master* ma numer zero. Tę wartość a posiadaja w obszarze 3 wszystkie wątki.

copyprivate(lista)

Przykład

```
!$OMP SINGLE read(1,*) a
!$OMP END SINGLE COPYPRIVATE(a)
```

Zmienna prywatna a jest przekazywana (broadcast) do innych wątków.

W przykadzie powyżej, pojedynczy wątek czyta z wejścia 1 wartość zmiennej a, a następnie przekazuje jej wartość pozostałym wątkom.

reduction(operator:lista)

Klauzula REDUCTION jest używana w obszarze gdzie dokonuje się składania (redukcji) zmiennej (np. sumowanie) i tylko w przypadku instrukcji redukujących. Zmienne listy powinny być typu SHARED dla tego obszaru. Dla każdego wątku tworzona jest prywatna kopia zmiennej tak, jak w przypadku PRIVATE. W końcu, zmienna dzielona jest redukowana z użyciem wartości jej prywatnych kopii z poszczególnych wątków…

Przykład

```
!$OMP DO
        do i = 1, 1000
          a=a+i
        end do
!$OMP END DO
```

Wynik sumowania nieokreślony!

```
!$OMP DO REDUCTION(+:a)
  do i = 1, 1000
          a=a+i
  end do
!$OMP END DO
```

Teraz jest Okay!

reduction(operator:lista)

Ogólna postać klauzuli redukcji dla wyrażeń typu

x = x operator wyrażenie
x = fun_wew(x, wyrażenie)

jest następująca:

REDUCTION(*operator*|*fun_wewnętrzna:lista*)

operator = +|*|-|.AND.|.OR.|.EQV.|.NEQV

fun_wewnętrzna = MAX|MIN|IAND|IOR|IEOR

Inne warunki...

- IF(skalarne_wyrażenie_logiczne)
- NUM_THREADS(skalarne_wyrażenie_całkowite)
- NOWAIT
- SCHEDULE(typ[, ilość])
- ORDERED

IF(wyrażenie)

Przykład

```
!$OMP PARALLEL IF(N >1000)
!$OMP DO
 do i = 1, N
   ...
 end do
!$OMP END DO
!$OMP END PARALLEL
```

Pętla do wykonuje się współbieżnie tylko wtedy gdy N jest większe niż 1000, w przeciwnym wypadku wykonywana jest przez jeden wątek.

NUM_THREADS(int)

- NUM_THREADS(N) - ustawia liczbę wątków dla danego obszaru obliczeń równolegych (np. jeśli zdefiniowaliśmy liczbę obszarów dyrektywą !$omp sections)

NOWAIT

- NOWAIT – pozwala zwolnić domyślną synchronizację wymuszona jakąś dyrektywą

- W przykładzie obok, druga pętla do nie ma nic wspólnego z pierwszą. Można więc przyspieszyć obliczenia, używając klauzuli NOWAIT w miejscu gdzie mamy domyślną synchronizację powodowaną dyrektywą !$OMP DO

Przykład

```
!$OMP PARALLEL
!$OMP DO
  do i = 1, 1000
    a=I
  enddo
!$OMP END DO NOWAIT
!$OMP DO
  do i = 1, 1000
    b=I
  enddo
!$OMP END DO
!$OMP END PARALLEL
```

SCHEDULE(typ[, chunk])

- Ogólnie

  schedule(type [,chunk])

  Określa w jaki sposób podzielić wykonanie pętli przez wątki

- schedule(static[,chunk])

  Iteracje dzielone są między wątki na porcje określone przez parametr chunk i kolejno wykonywane wg numerów wątków.

- Schedule(dynamic[,chunk])

  Iteracje dzielone są między wątki na porcje określone przez parametr chunk i przydzielane kolejno wolnym wątkom.

- schedule(guided[,chunk])

  Porcje przydzielane wątkom tak, że ileść określona przez chunk zmniejsza się eksponencjalnie.

ORDERED ...

- Zadanie 1. Opisać klauzulę ORDERED. Podać przykłady.
- Zadanie 2. Narysować schematy dla klauzuli SCHEDULE w trzech omawianych przypadkach: static, dynamic oraz guided.
- OK! To tyle na dziś...

Klauzule w dyrektywach

KLAUZULA / pragma	PARALLEL	DO /for	SECTIONS	SINGLE	PARALLEL DO / for	PARALLEL SECTIONS	PARALLEL WORKSHARE
if	X				X	X	X
private	X	X	X	X	X	X	X
shared	X				X	X	X
firstprivate	X	X	X	X	X	X	X
lastprivate		X	X		X	X	
default	X				X	X	X
reduction	X	X	X		X	X	X
copyin	X				X	X	X
copyprivate				X			
ordered		X			X		
schedule		X			X		
nowait		X	X	X			
num_threads	X				X	X	X

Problemy...?