

Paralleles Programmieren mit OpenMP und MPI

OpenMP-Übungsaufgaben

Hartmut Häfner

STEINBUCH CENTRE FOR COMPUTING - SCC

Parallele Berechnung von PI

```
program compute_pi
integer                :: i
integer, parameter    :: n=500000000, dp = kind(1.d0)
real(kind=dp)         :: w,x,sum,pi,d

w=1.0/n; sum=0.0
!$OMP PARALLEL PRIVATE(x,d), SHARED(w,sum)
!$OMP DO REDUCTION(+: sum)
do i=1,n
    x = (i-0.5) * w
    d = w * SQRT(1.0 - x**2)
    sum = sum + d
enddo
!$OMP END DO
!$OMP END PARALLEL
pi = 4. * sum
print *, 'computed pi = ', pi
end program compute_pi
```

bwUniCluster

1 core:

real	1.64s
user	1.64s

2 cores: (Sp = 1.95)

real	0.84s
user	1.67s

4 cores: (Sp = 3.81)

real	0.43s
user	1.70s

8 cores: (Sp = 7.13)

real	0.23s
user	1.81s

16 cores: (Sp = 10.25)

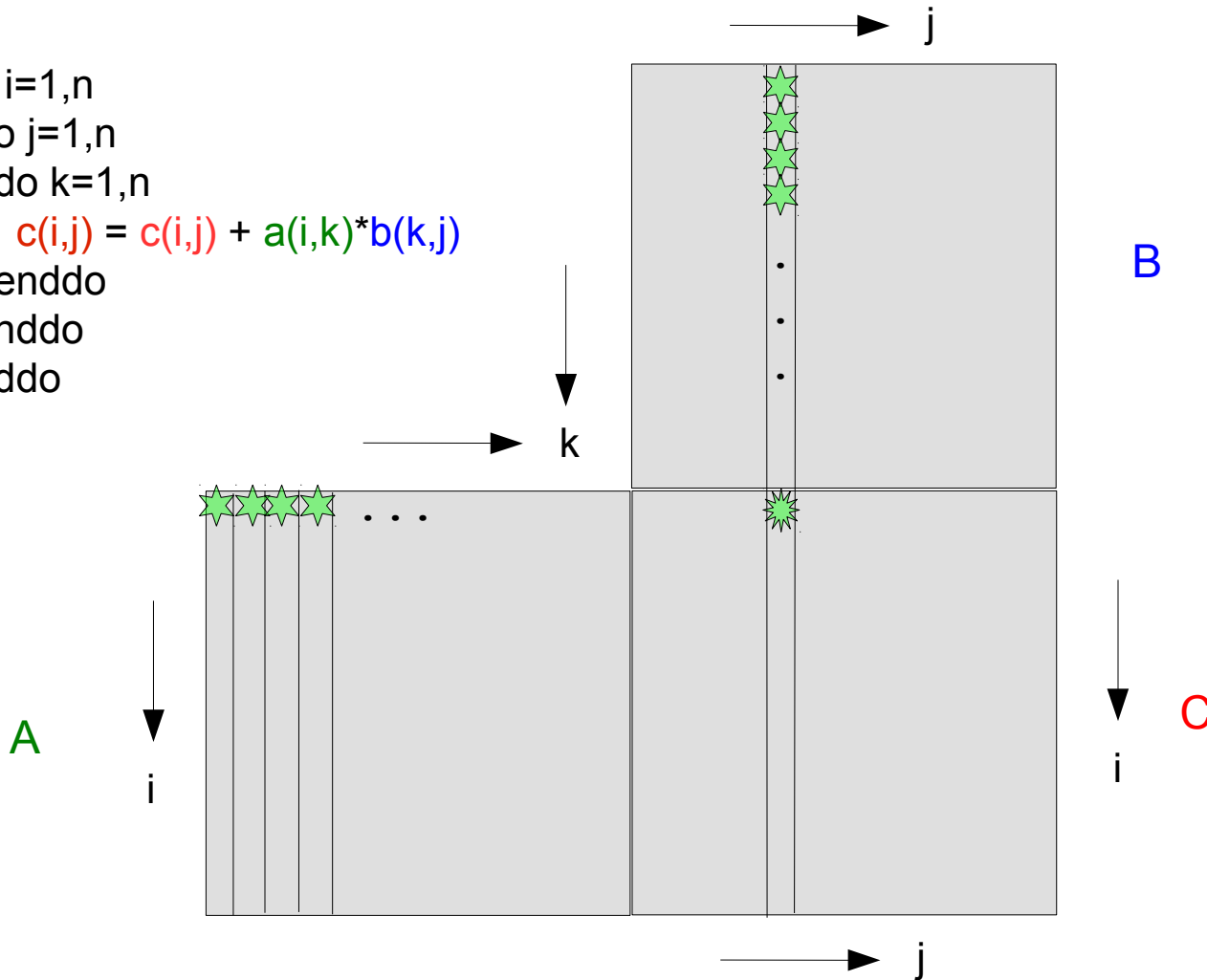
real	0.16s
user	2.38s

```
module add compiler/intel
cp /work/kit/scc/ku8089/OpenMP-Uebung/pi.f90 .
(cp /work/kit/scc/ku8089/OpenMP-Uebung/seconds.c .)
(icc -c -O -DFTNLINKSUFFIX seconds.c)
ifort -O3 -o pi pi.f90 seconds.o      bzw.
ifort -O3 -openmp -o pi_par pi.f90 seconds.o
./pi      bzw.
export OMP_NUM_THREADS=4
./pi_par
cp /work/kit/scc/ku8089/OpenMP-Uebung/jobuc_omp.sh .
msub -l ADVRES=kit-workshop.200 ./jobuc_omp.sh
```

Schreiben Sie das parallele Programm PI so um, dass Sie **ohne die Klausel `reduction`** auskommen!

Die Matrizenmultiplikation $C = A * B$

```
do i=1,n
  do j=1,n
    do k=1,n
       $c(i,j) = c(i,j) + a(i,k)*b(k,j)$ 
    enddo
  enddo
enddo
```



```
ifort -O3 -openmp -o mmul_ijk mmul_ijk.f90 seconds.o  
export OMP_NUM_THREADS=4  
./mmul_ijk      oder  
msub -l ADVRES=kit-workshop.200 jobuc_ompmul.sh
```

Parallelisieren Sie die Matrizenmultiplikation MMUL_IJK!

Stellen Sie um auf die JKI-Form der Matrizenmultiplikation und parallelisieren Sie diese! (Sie können auch die Umgebungsvariable OMP_SCHEDULE aktivieren in dem Skript jobuc_ompmul.sh!)

Der Gaußalgorithmus $A \cdot x = b$

```
do j=1,n-1
  pivot = 1/a(j,j)
  do i=j+1,n
    a(i,j) = a(i,j) * pivot
  enddo
```

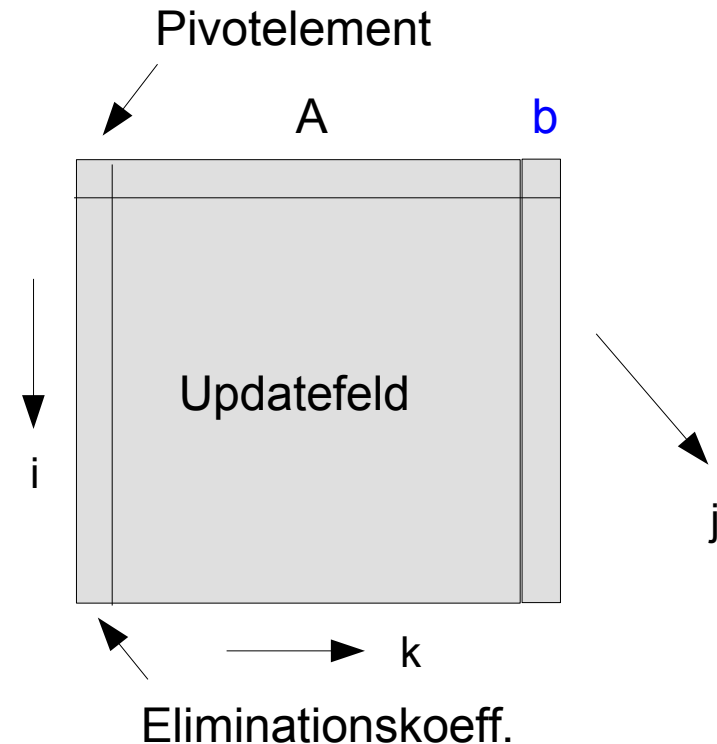
! Berechnung der
! Elim. koeff.

```
do k=j+1,n
  do i=j+1,n
    a(i,k) = a(i,k) - a(i,j)*a(j,k)
  enddo
enddo
```

! Update
! der
! Restmatrix

```
do i=j+1,n
  b(i) = b(i) - a(i,j)*b(j)
enddo
enddo
```

! Update
! der
! rechten Seite



```
cp /work/kit/scc/ku8089/OpenMP-Uebung/gauss.f90 .  
(cp /work/kit/scc/ku8089/OpenMP-Uebung/seconds.c .)  
(icc -c -O -DFTNLINKSUFFIX seconds.c)  
ifort -O3 -o gauss gauss.f90 seconds.o      bzw.  
ifort -openmp -O3 -o gauss_par gauss.f90 seconds.o  
./gauss      bzw.  
export OMP_NUM_THREADS=4  
./gauss_par  
msub -l ADVRES=kit-workshop.200 jobuc_ompgauss.sh
```

Parallelisieren Sie den Gauß-Algorithmus GAUSS!
Optimieren Sie den Gauß-Algorithmus für NUMA-Architekturen.