Karlsruhe Institute of Technology

Parallel Programming with MPI and OpenMP

OpenMP-Exercises

Steinbuch Centre for Computing

Forschungszentrum Karlsruhe
in der Helmholtz-Gemeinschaft

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Parallel Computation of PI

```
program compute_pi
integer                         :: i
integer, parameter              :: n=50000000, 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
```

Copying, Compiling and Starting of PI

```
cp /work/kit/scc/ku8089/OpenMP-Exercise/pi.f90  .
(cp /work/kit/scc/ku8089/OpenMP-Exercise/seconds.c  .)
(icc -c -O -DFTNLINKSUFFIX seconds.c)
ifort -O3 -o pi pi.f90 seconds.o     or
ifort -O3 -qopenmp -o pi_par pi.f90 seconds.o
./pi  or
export $OMP_NUM_THREADS=4
./pi_par  or
msub -l advres=gridKA-MPI.38 ./jobuc_omp.sh
```

Rewrite the parallel program PI so that the clause `reduction` is not used!

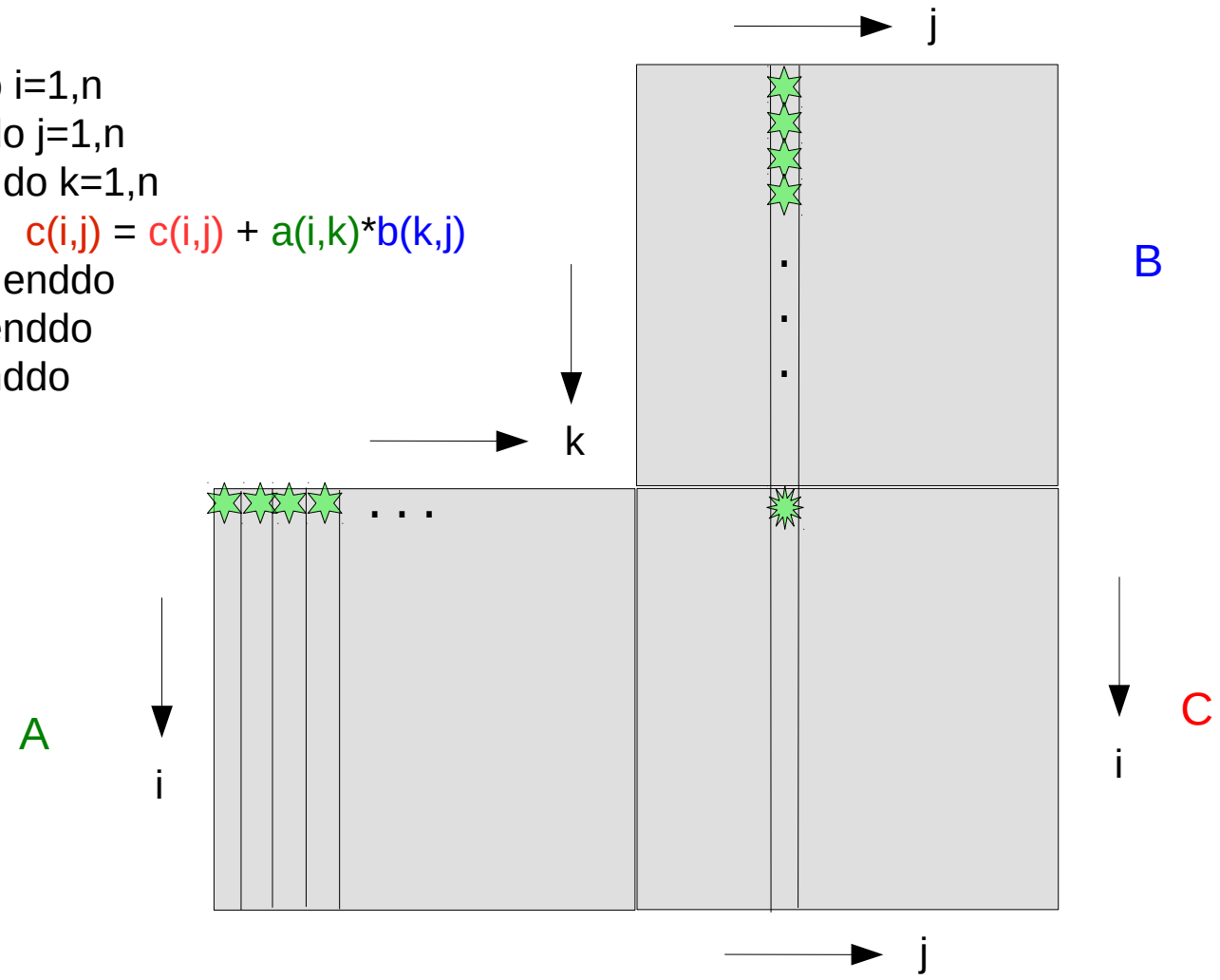
Parallel Computation of PI with `atomic` (*update*)

- `atomic` is a special case of a critical section, that is used to assign a new value to scalar variables in simple assignments.
- The *assignment statement* must have the following syntax:
  x = x *operator* expr; x = *intrinsic*(x,expr)

```
program compute_pi
integer                :: i
integer, parameter     :: n=500000000, dp = kind(1.d0)
real(kind=dp)          :: w,x,pi,d,sum,glob_sum
w=1.d0/n; glob_sum=0.d0
!$OMP PARALLEL PRIVATE(x,d,sum)
sum = 0.d0
!$OMP DO
do i=1,n
  x = (i-0.5)*w; d = w*SQRT(1.0-x**2); sum = sum+d
enddo
!$OMP ATOMIC
glob_sum = glob_sum + sum
!$OMP END PARALLEL
pi = 4. * glob_sum
end program compute_pi
```

The Matrix Multiplication 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
```



Copying, Compiling and Starting of MMUL_IJK

```
ifort -O3 -qopenmp -o mmul_ijk_omp mmul_ijk.f90 seconds.o
export OMP_NUM_THREADS=4
./mmul_ijk_omp      or
msub -l advres=gridKA-MPI.38 jobuc_ompmmul.sh
```

Parallelize the matrix multiplication MMUL_IJK!

Parallel Columnwise Matrix Multiplication

```
program mmul_jki
integer, parameter :: dp=kind(1.d0)
real(kind=dp), dimension(:,:), allocatable:: a, b, c
. . .

!$OMP PARALLEL SHARED(a, b, c)
!$OMP DO
do j = 1, n
  do k = 1, n
    do i = 1, n
      a(i,j) = a(i,j) + b(i,k) * c(k,j)
    end do
  end do
end do
!$OMP END DO
!$OMP END PARALLEL
. . .
```

bwUniCluster with n=4000

1 core (serial):

```
real      10.1s
user      10.1s
```

2 cores:

```
real       7.0s
user      14.0s
```

4 cores:

```
real       3.5s
user      14.0s
```

8 cores:

```
real       1.9s
user      15.1s
```

16 cores:

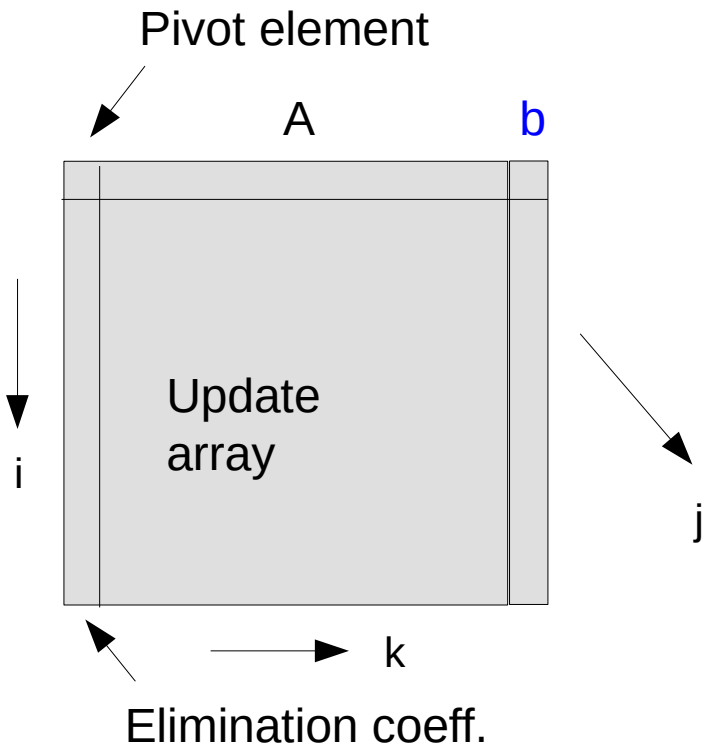
```
real       1.1s
user      17.0s
```

Gauss Algorithm A*x = b

```
do j=1,n-1
  pivot = 1/a(j,j)
  do i=j+1,n                    ! Computation of
    a(i,j) = a(i,j) * pivot     ! Elim. coeff.
  enddo

  do k=j+1,n
    do i=j+1,n                      ! Update
      a(i,k) = a(i,k) – a(i,j)*a(j,k)  ! of
    enddo                           ! update array
  enddo

  do i=j+1,n                    ! Update
    b(i) = b(i) – a(i,j)*b(j)   ! of
  enddo                         ! right hand side
enddo
```



Copying, Compiling and Starting of the Gauss Alg.

```
cp /work/kit/scc/ku8089/OpenMP-Exercise/gauss.f90   .
(cp /work/kit/scc/ku8089/OpenMP-Exercise/seconds.c   .)
(icc -c -O -DFTNLINKSUFFIX seconds.c)
ifort -O3 -o gauss gauss.f90 seconds.o      bzw.
ifort -qopenmp -O3 -o gauss_par gauss.f90 seconds.o
./gauss       or
export OMP_NUM_THREADS=4
./gauss_par
msub -l advres=gridKA-MPI.38 jobuc_ompgauss.sh
```

Parallelize the Gauss algorithm!
Optional: optimize the Gauss algorithm for NUMA-architectures.