Karlsruhe Institute of Technology

Parallel Programming with MPI and OpenMP

MPI-Exercises

Hartmut Häfner, Steinbuch Centre for Computing (SCC)

STEINBUCH CENTRE FOR COMPUTING - SCC

KIT – University of the State of Baden-Württemberg and
National Laboratory of the Helmholtz Association

Sum over all Processors

Create a MPI-program with the following simple structure:
Process 0 sends a message to process 1,
Process 1 sends a message to process 2,
…
Process N-1 sends a message to process 0.

The message is a 64-bit real variable called sum, that is initialized with value zero by process 0. Each process adds its process-Id to variable sum and sends the message to the next process. At the end process 0 prints the result.

```
MPI_Init(&argc, &argv)     MPI_Finalize()
MPI_Comm_rank(MPI_Comm_world, &rank)
MPI_Comm_size(MPI_Comm_world, &size)
MPI_Send(buf, count, datatype, dest, tag, comm)
MPI_Recv(buf, count, datatype, source, tag, comm, status)
```

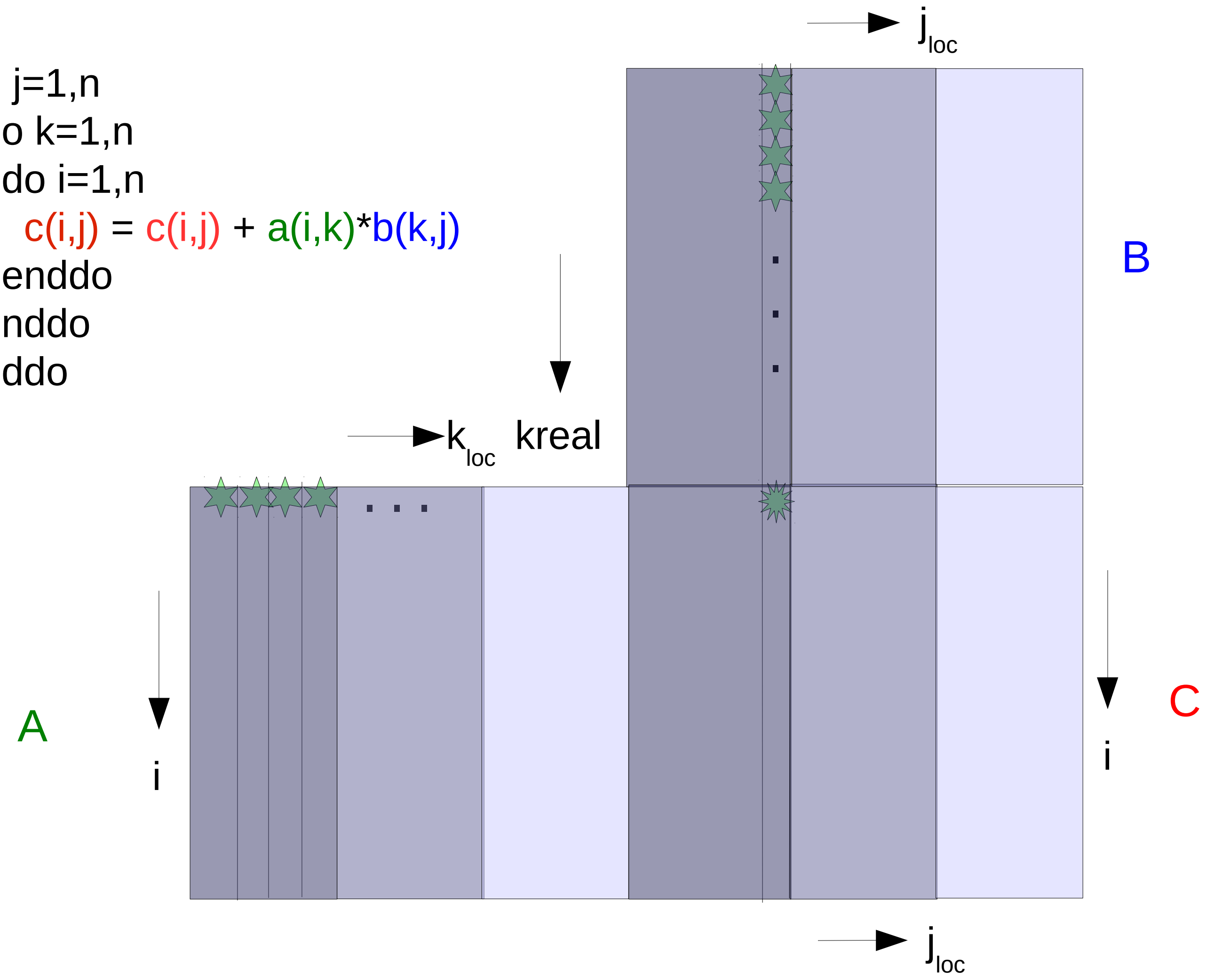
Copying, Compiling and Starting of sum_template.c

```
(module av)
module add mpi/openmpi/2.1-intel-16.0
cp /work/kit/scc/ku8089/MPI-Exercise/mpi_sum_template.c  .
mpicc -O3 -o mpi_sum_simple mpi_sum_template.c
mpirun -np 4 ./mpi_sum_simple
msub -q multinode -l advres=gridKA-MPI.38 jobuc_ompi.sh
```

Optimize the program so that the number of necessary communication slots drops to O($\log_2$n). The program must only run on processor number with power of 2.

The Parallel Matrix Multiplication C = A * B

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



Copying, Compiling and Starting of PARMMUL

```
cp /work/kit/scc/ku8089/MPI-Exercise/parmmul_standard.f90  .
cp /work/kit/scc/ku8089/MPI-Exercise/seconds.c  .
icc -c -O -DFTNLINKSUFFIX seconds.c
mpif90 -O3 -o parmmul parmmul_standard.f90 seconds.o

mpirun -np 4 ./parmmul       or
msub -q multinode -l advres=gridKA-MPI.38 jobuc_ompi.sh
```

The parallel matrix multiplication PARMMUL_STANDARD uses blockwise parallelization with „standard“ Send and blocking Receive.

a) Rewrite the program so that it works for all matrix sizes.

b) Rewrite the program so that it runs on one processor.

PARMMUL-Code

```
program PARMMUL

integer, parameter ::   n = 100
real*8, parameter ::   one = 1.0, eps = 1.d-10
real*8, pointer ::  a(:,:), ap(:,:), b(:,:), c(:,:)
real*8  t0, t0e, t1, t1e
integer, pointer ::  s(:), kbs(:), kbe(:)
integer ib, lrank, p, rank, totid, frtid, sid1, sid2, rid1, rid2, err
include 'mpif.h'
integer istat(MPI_STATUS_SIZE)

!-------------------------
!**** Communication setup |
!-------------------------
call MPI_INIT(err)
call MPI_COMM_RANK(MPI_COMM_WORLD,rank,err)
call MPI_COMM_SIZE(MPI_COMM_WORLD,p,err)

ns = n/p
allocate(a(n,ns+1), ap(n,ns+1), b(n,ns+1), c(n,ns+1), STAT = err)
allocate(s(p), kbs(p), kbe(p), STAT = err)

!------------------------------------------
!**** Installation of a ring communication |
!------------------------------------------
totid = rank + 1
if (totid == p) totid = 0
```

PARMMUL-Code (2)

```
frtid = rank - 1
if (frtid < 0) frtid = p - 1
lrank = rank + 1               ! 1 <= lrank <= p

do ib=1,MOD(n,p)
  s(ib) = ns + 1               ! s is blockwidth
enddo

do ib=MOD(n,p)+1,p
  s(ib) = ns
enddo

kbs(1) = 1                     ! kbs ist pointer to beginning of blocks
kbe(1) = s(1)                  ! kbe ist pointe to end of blocks

do ib=2,p
  kbs(ib) = kbs(ib-1) + s(ib-1)
  kbe(ib) = kbs(ib) + s(ib) – 1
enddo

do k=1,s(lrank)
  do i=1,n
    a(i,k) = one
    b(i,k) = DBLE(i)
  enddo
enddo
```

PARMMUL-Code (3)

```
call SECONDS(t0,t0e)

do itact=1,p
  ib  = MOD(lrank-itact+p,p) + 1  ! Block number in cycle itakt
  ibn = MOD(ib-2+p,p) + 1         ! Block number of data to be received

!------------------------------------------------------------------
!**** Standard SEND of A with m-type 10                             |
!------------------------------------------------------------------
  call MPI_SEND(a(1,1),n*s(ib),MPI_REAL8,totid,10,MPI_COMM_WORLD,err)

  do j=1,s(lrank)                 ! MMUL C = A * B
    do k=1,s(ib)
      kreal = kbs(ib) + k - 1
      if ((k == 1) .and. (itact == 1)) then
        do i=1,n
          c(i,j) = a(i,k)*b(kreal,j)
        enddo
      else
        do i=1,n
          c(i,j) = c(i,j) + a(i,k)*b(kreal,j)
        enddo
      endif
    enddo
  enddo
```

PARMMUL-Code (4)

```
!--------------------------------------------------------
!**** Standard RECV of A with m-type 10                  |
!--------------------------------------------------------
  call MPI_RECV(a(1,1),n*s(ibn),MPI_REAL8,frtid,10,&
               &MPI_COMM_WORLD,istat,err)
enddo
call SECONDS(t1,t1e)

do j=1,s(lrank)
  do i=1,n
    if (ABS(c(i,j)-(n*(n+1)/2.d0)) > eps) then
      print *,' Matrix C is wrong'
    endif
  enddo
enddo
if (lrank == 1) then
  print *,' N=',n,'  Time [sec]=',t1e-t0e,&
         &'  MFLOPS=',2.*n*n*n*1.e-6/(t1e-t0e)
endif

call MPI_FINALIZE(err)
end
```