Karlsruhe Institute of Technology

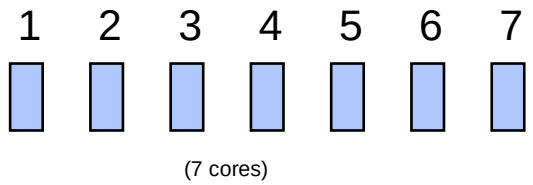
Parallel Programming with MPI and OpenMP

MPI-Solutions

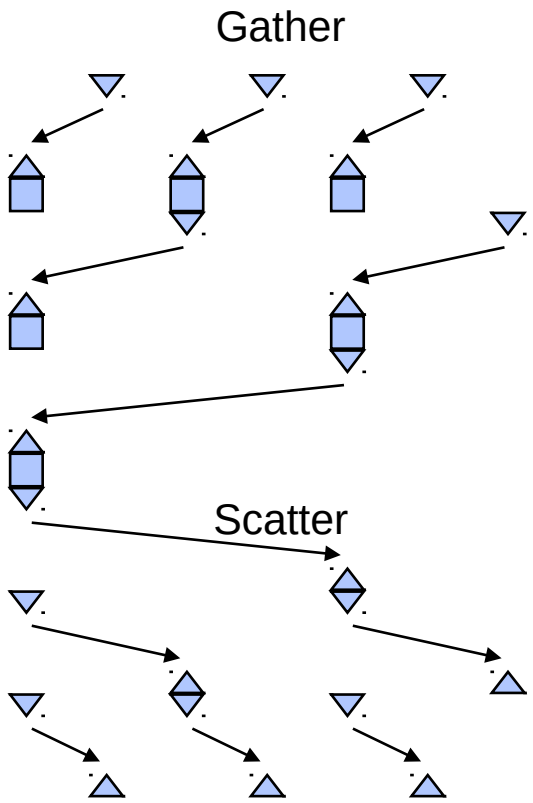
Hartmut Häfner, Steinbuch Centre for Computing (SCC)

STEINBUCH CENTRE FOR COMPUTING - SCC

Sum over all Processors



(7 cores)



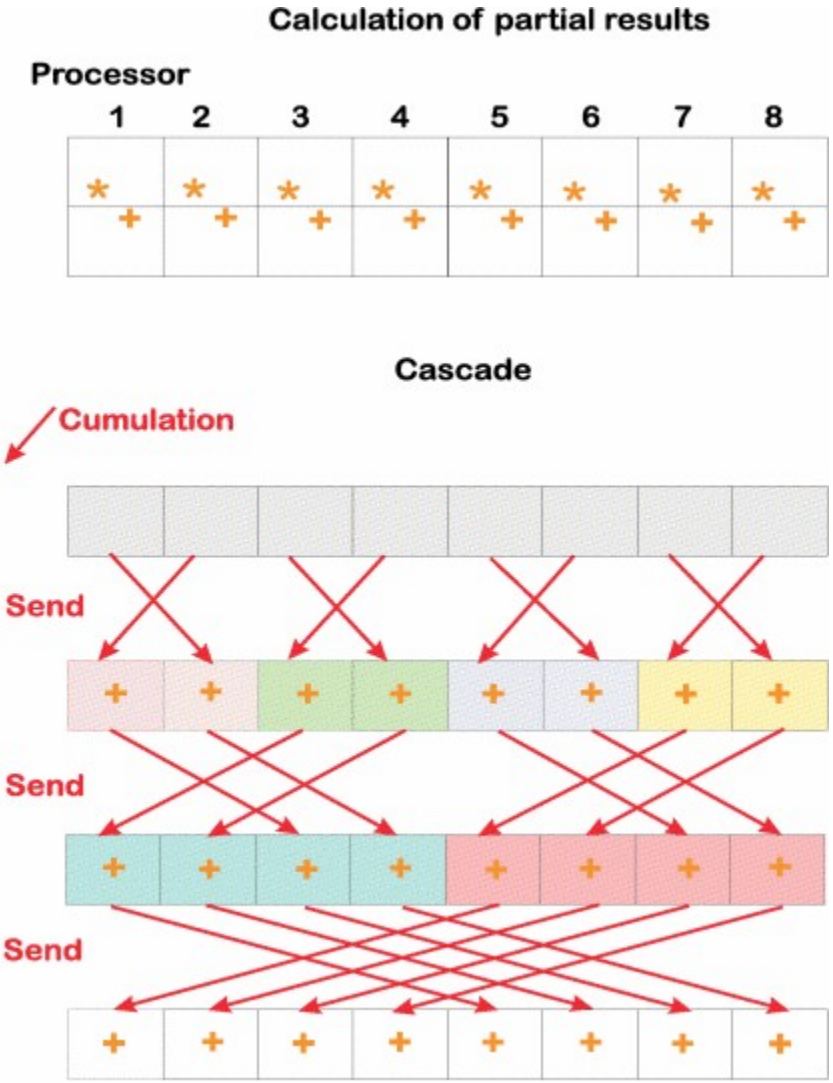
means vectorized calculation of local scalars like local Min/Max or local dot products on the locally stored vector parts

means receiving scalars

means sending scalars

means scalar Min/Max - or dot products calculation of local and received scalars

Sum over all processors (optimized)



PARMMUL-Code (with alternating buffers)

```
      program PARMMUL

      integer, parameter ::   n = 1000
      real*8, parameter ::   one = 1.0
      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 ist blockwidth
enddo

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

kbs(1) = 1                      ! kbs is pointer to beginning of blocks
kbe(1) = s(1)                   ! kbe is pointer 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,INT(p/2)
        ib = MOD(lrank-2*itact+p+1,p) + 1  ! Block number in cycle itakt
        ibn = MOD(ib-2+p,p) + 1            ! Block number of data to be received
!-----------------------------------------------------------------------
!**** Asynchronous RECV of AP with m-type 10 and sync-variable rid1 |
!-----------------------------------------------------------------------
        call MPI_IRECV(ap(1,1),n*s(ibn),MPI_REAL8,frtid,10,
     *                 MPI_COMM_WORLD,rid1,err)
!-----------------------------------------------------------------------
!**** Asynchronous SEND of A with m-type 10 and sync-variable sid1 |
!-----------------------------------------------------------------------
        call MPI_ISEND(a(1,1),n*s(ib),MPI_REAL8,totid,10,
     *                 MPI_COMM_WORLD,sid1,err)
!-------------------------------------------------------
!**** WAIT-SEND (with m-type 20) and sync-variable sid2 |
!-------------------------------------------------------
        if (itact /= 1) then
          call MPI_WAIT(sid2,istat,err)
        endif
```

PARMMUL-Code (4)

```
        do j=1,s(lrank)                  ! MMUL with A in phase 1 of
          do k=1,s(ib)                   ! iteration itakt
            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

        ib = MOD(lrank-2*itact+p,p) + 1
        ibn = MOD(ib-2+p,p) + 1
!-------------------------------------------------------
!**** WAIT-RECV (with m-type 10) and sync-variable rid1 |
!-------------------------------------------------------
        call MPI_WAIT(rid1,istat,err)
!-------------------------------------------------------------------
!**** Asynchronous RECV of A with m-type 20 and sync-variable rid2 |
!-------------------------------------------------------------------
        call MPI_IRECV(a(1,1),n*s(ibn),MPI_REAL8,frtid,20,
     *                 MPI_COMM_WORLD,rid2,err)
```

PARMMUL-Code (5)

```
!-------------------------------------------------------------------
!**** Asynchronous SEND of AP with m-type 20 and sync-variable sid2 |
!-------------------------------------------------------------------
       call MPI_ISEND(ap(1,1),n*s(ib),MPI_REAL8,totid,20,
     *                MPI_COMM_WORLD,sid2,err)
!-----------------------------------------------------------
!**** WAIT-SEND (with m-type 10) and sync-variable sid1 |
!-----------------------------------------------------------
       call MPI_WAIT(sid1,istat,err)
       do j=1,s(lrank)                          ! MMUL with AP in phase 2 of
         do k=1,s(ib)                           ! iteration itakt
           kreal = kbs(ib) + k - 1
           do i=1,n
             c(i,j) = c(i,j) + ap(i,k)*b(kreal,j)
           enddo
         enddo
       enddo
!-----------------------------------------------------------
!**** WAIT-RECV (with m-type 20) and sync-variable rid2 |
!-----------------------------------------------------------
       call MPI_WAIT(rid2,istat,err)
     enddo
     call SECONDS(t1,t1e)
     do j=1,s(lrank); do i=1,n; if (c(i,j) /= n*(n+1)/2.) then
       print *,' Matrix C is wrong'; endif; enddo; enddo
     if (lrank == 1) then
       print *,' Time [sec]=',t1e-t0e,' MFLOPS=',2.*n*n*n*1.e-6/(t1e-t0e)
     endif
     call MPI_FINALIZE(err)
     end
```