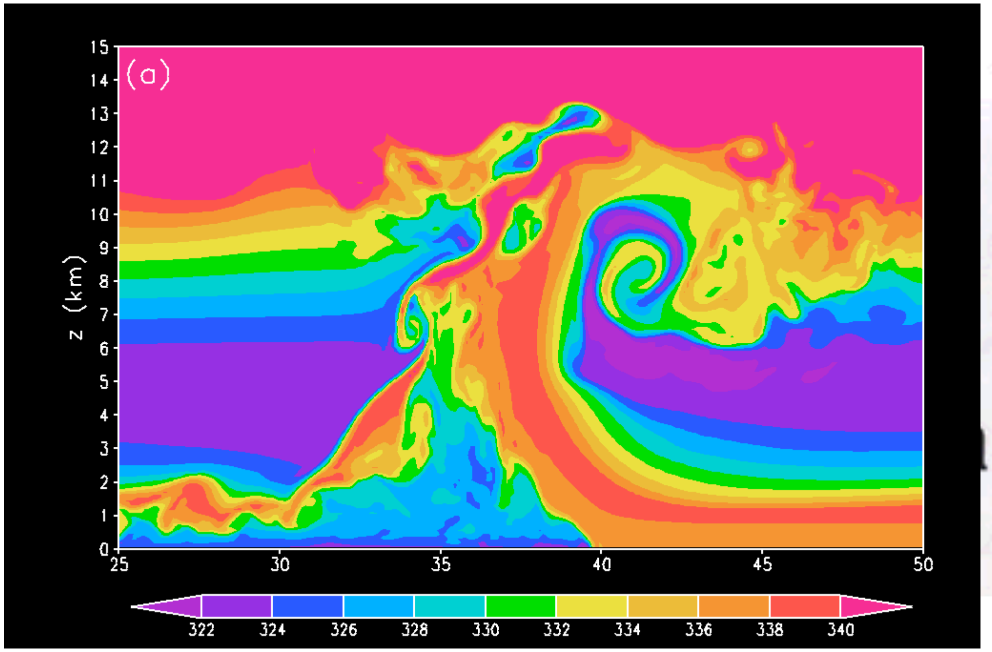
Parallelization in CM1

George Bryan
NCAR

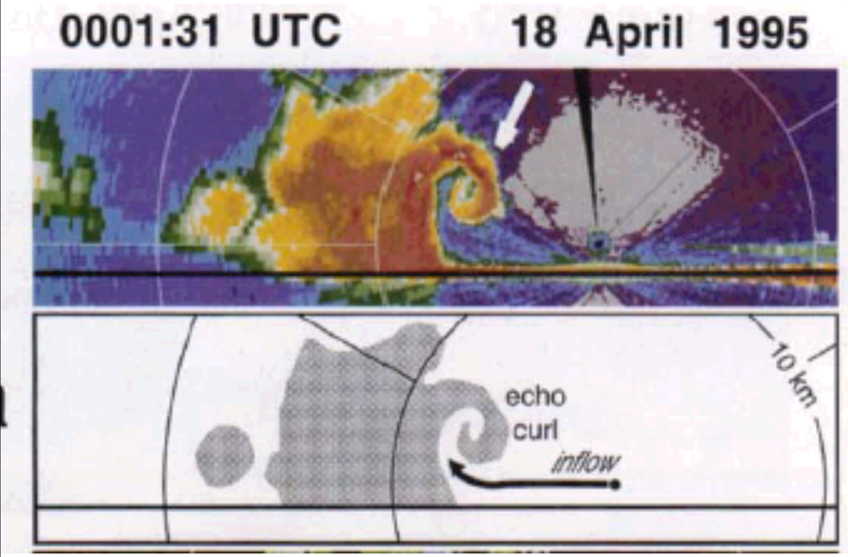
Presentation at NCSA
2 December 2009

Frequently Asked Questions:

- What is CM1?
  - A 3d nonhydrostatic atmospheric model developed for idealized modeling of clouds/cloud-systems at LES scales (Δ ≈ 10-100 m)
  - Specifically designed for distributed-memory computing systems



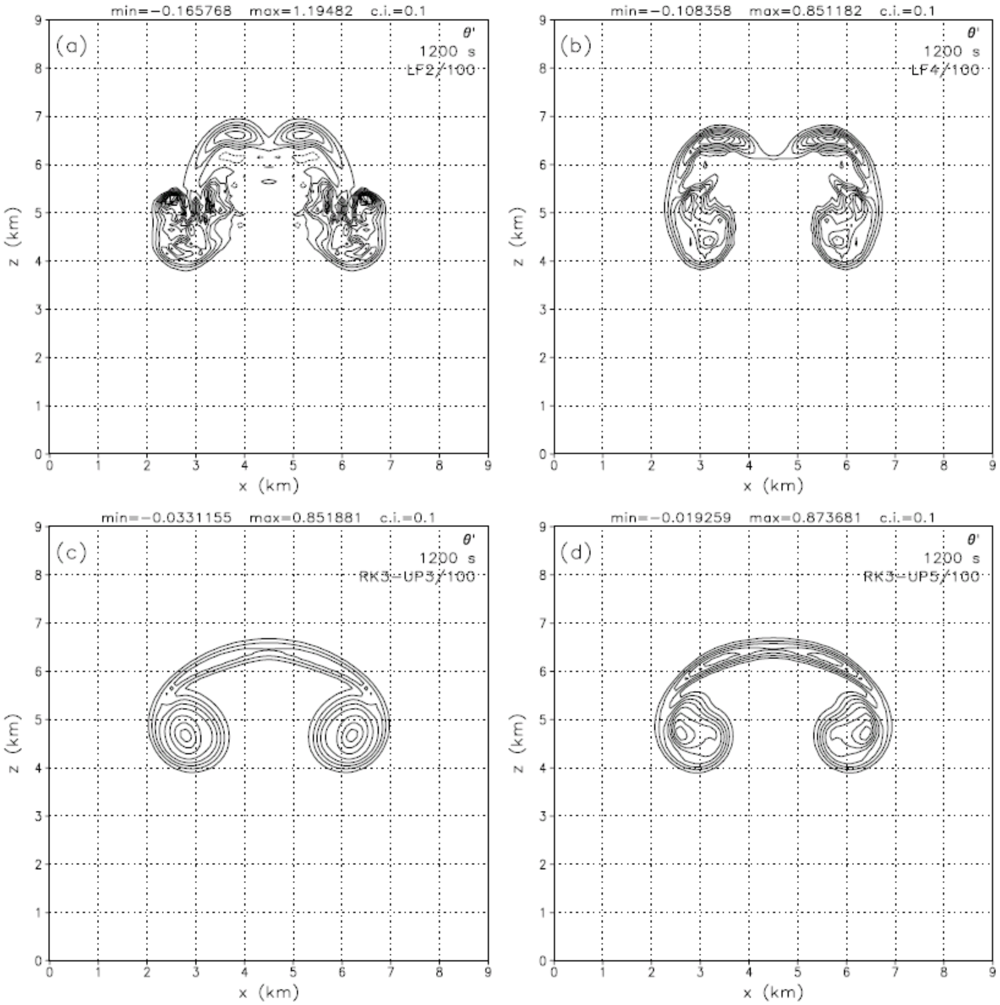
Bryan (2002)



Wakimoto et al. (1996)

Frequently Asked Questions:

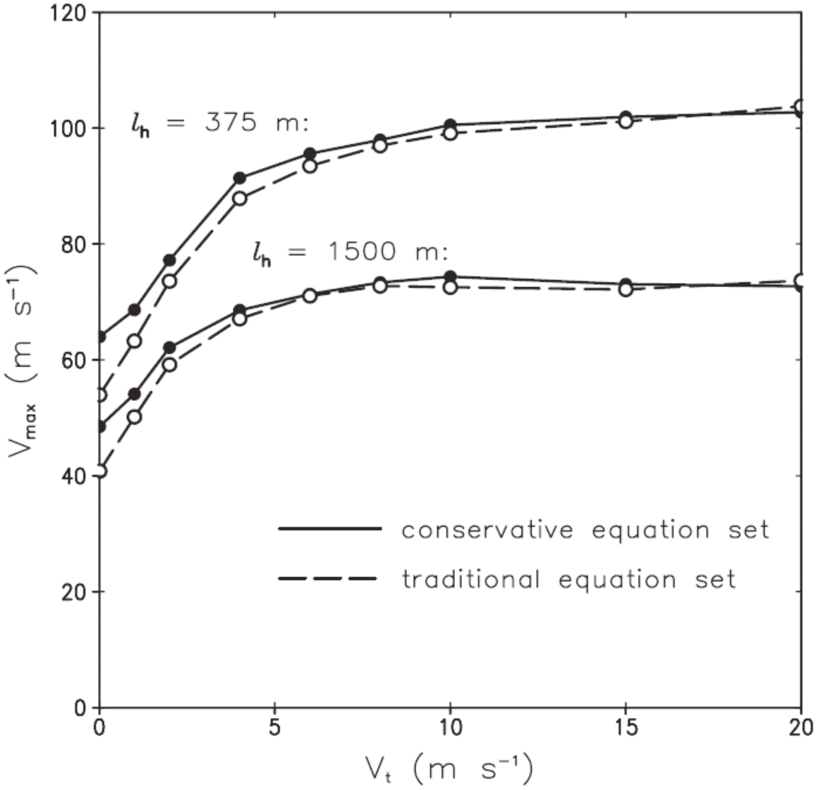
- Why CM1?
- (or, why not WRF, ARPS?)
  - CM1 was “born” in the late 90s at Penn State as a modified version of MM5 (fifth-generation mesoscale model → first-generation cloud model)
  - Primary solver is similar to WRF (ARW) … RK3 split-explicit, 5th/6th-order advection … but uses a Cartesian height coordinate



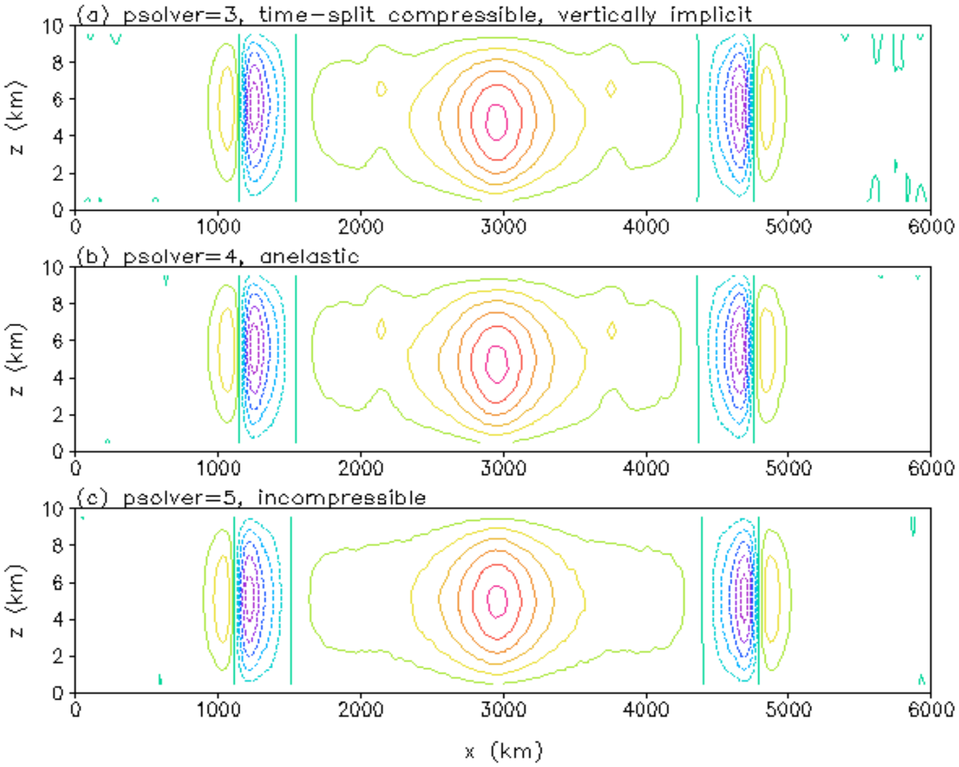
Bryan (2002)

Frequently Asked Questions:

- What makes CM1 different from other models?
  - Energy conservation: considers heat content of liquid/solid water, includes dissipative heating (as of cm1r12)
  - Five pressure solvers: (incompressible, anelastic, 3 compressible)



Bryan and Rotunno (2009, MWR)



www.mmm.ucar.edu/people/bryan/cm1

Frequently Asked Questions:

- How fast is CM1?
  - Depends:
  - With no terrain, it’s very fast (roughly twice as fast as ARW)
  - Using energy-conserving equations adds 5-15%

- What parallelization options are available in CM1?
  - Shared memory parallelization with OpenMP is available in CM1 …. but hasn’t been developed much
  - Distributed memory parallelization using MPI … focus of this talk
  - Can do hybrid OpenMP / MPI

Distributed-memory parallelization in CM1

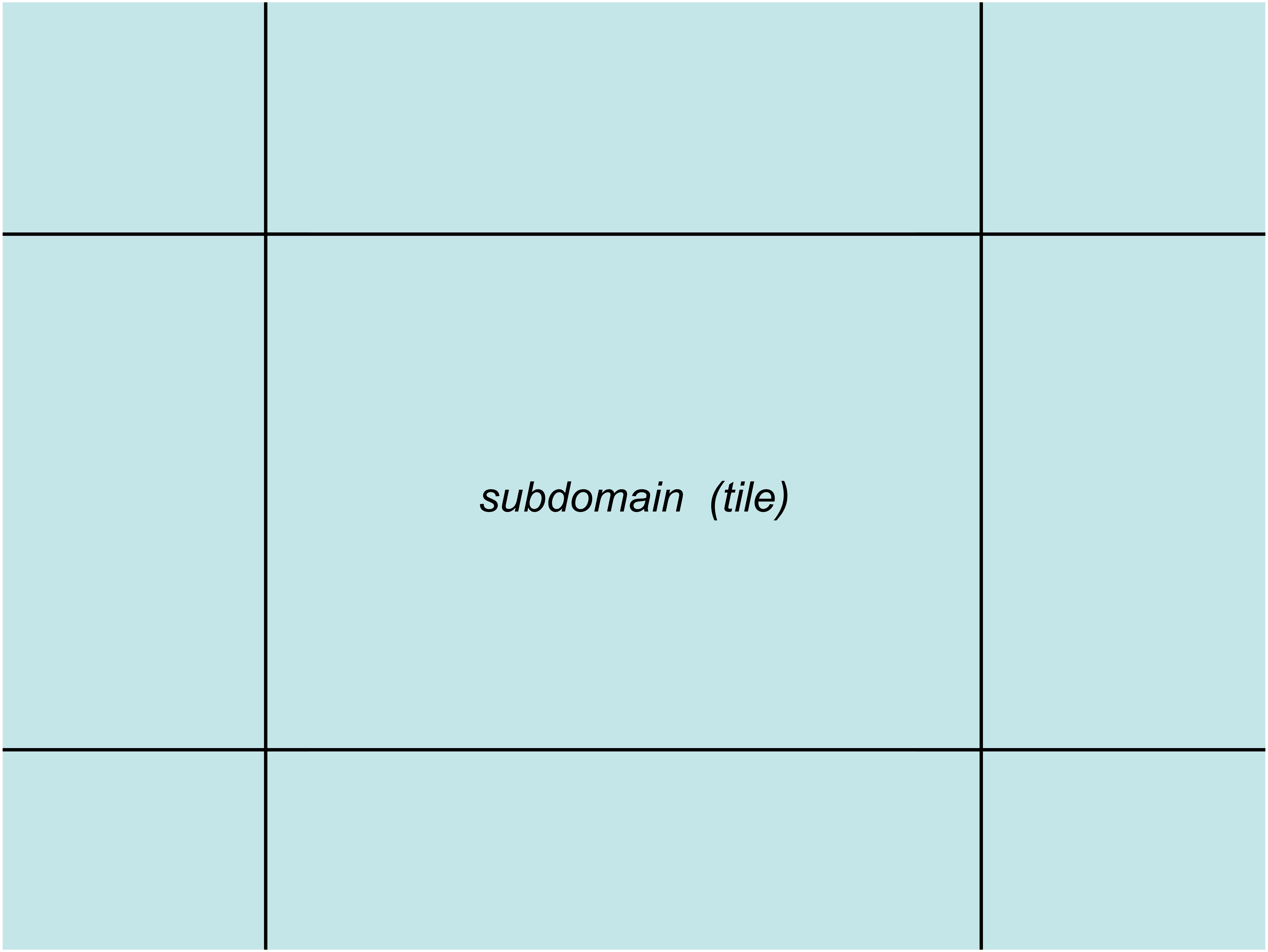
- 2d domain decomposition:
  (example using 12 processors)

myid = 8	myid = 9	myid = 10	myid = 11
myid = 4	myid = 5	myid = 6	myid = 7
myid = 0	myid = 1	myid = 2	myid = 3

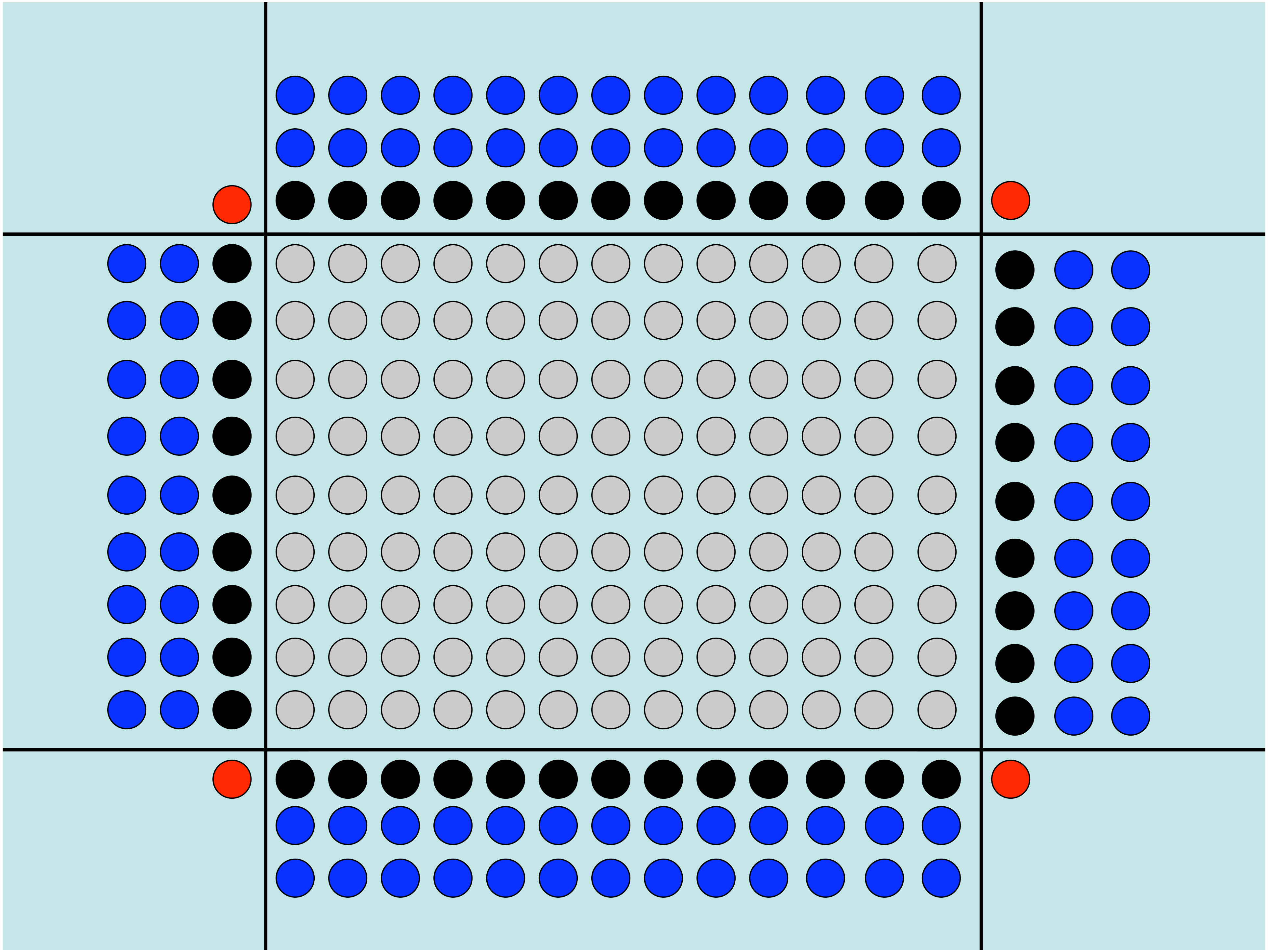
Distributed-memory parallelization in CM1

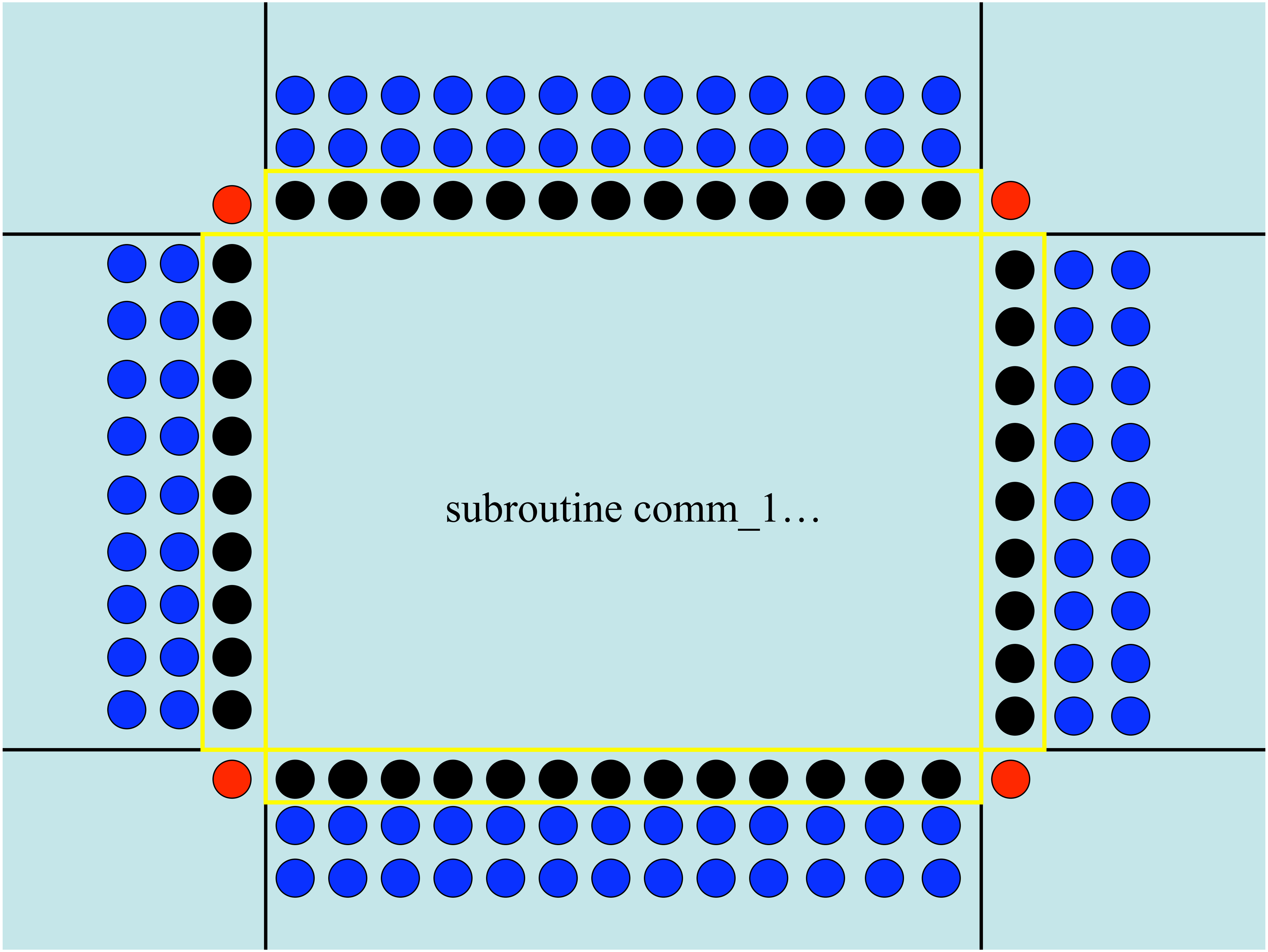
- 2d domain decomposition:
  (example using 12 processors)

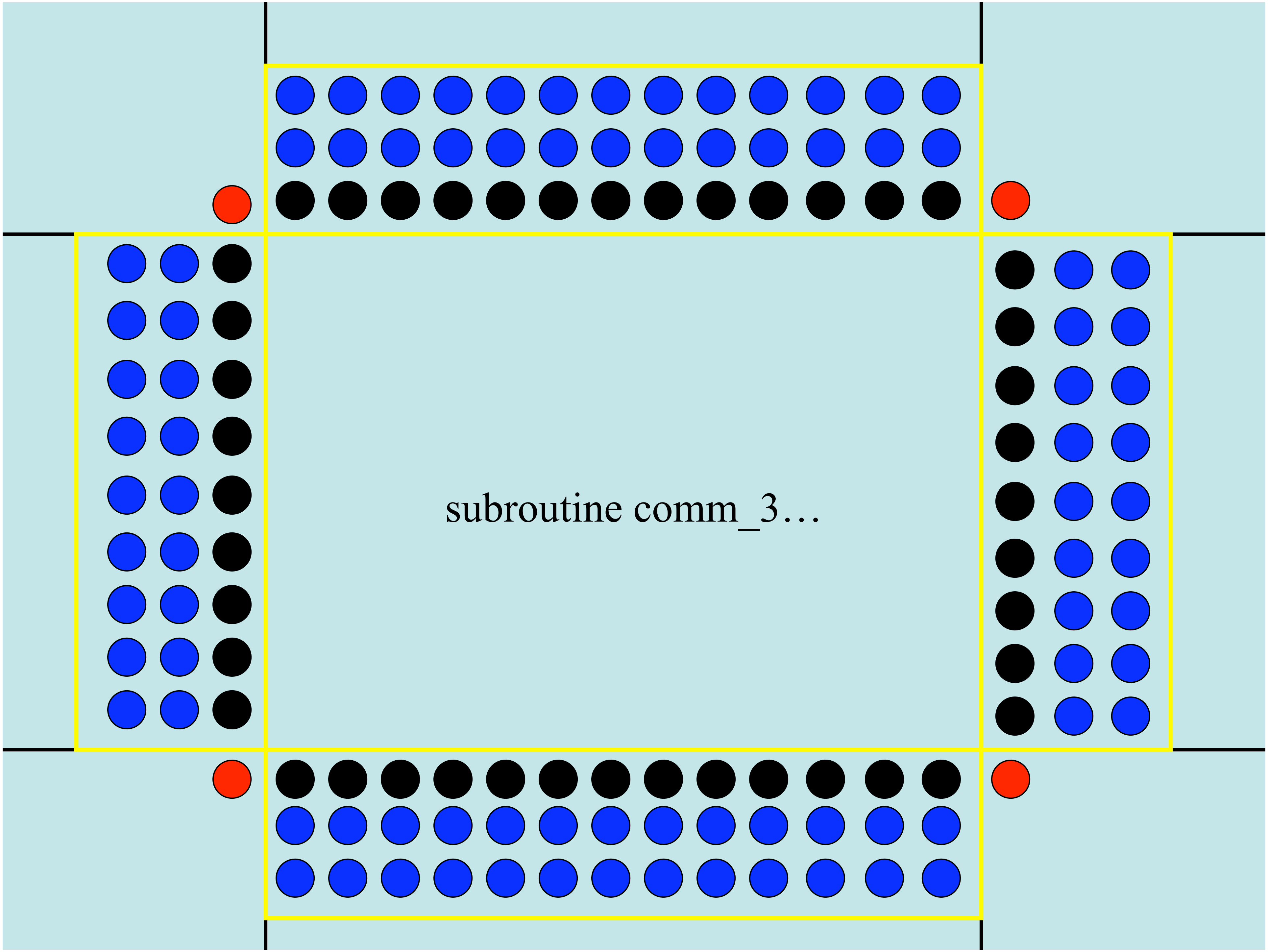
myj = 3	myid = 8	myid = 9	myid = 10	myid = 11
myj = 2	myid = 4	myid = 5	myid = 6	myid = 7
myj = 1	myid = 0	myid = 1	myid = 2	myid = 3
	myi = 1	myi = 2	myi = 3	myi = 4

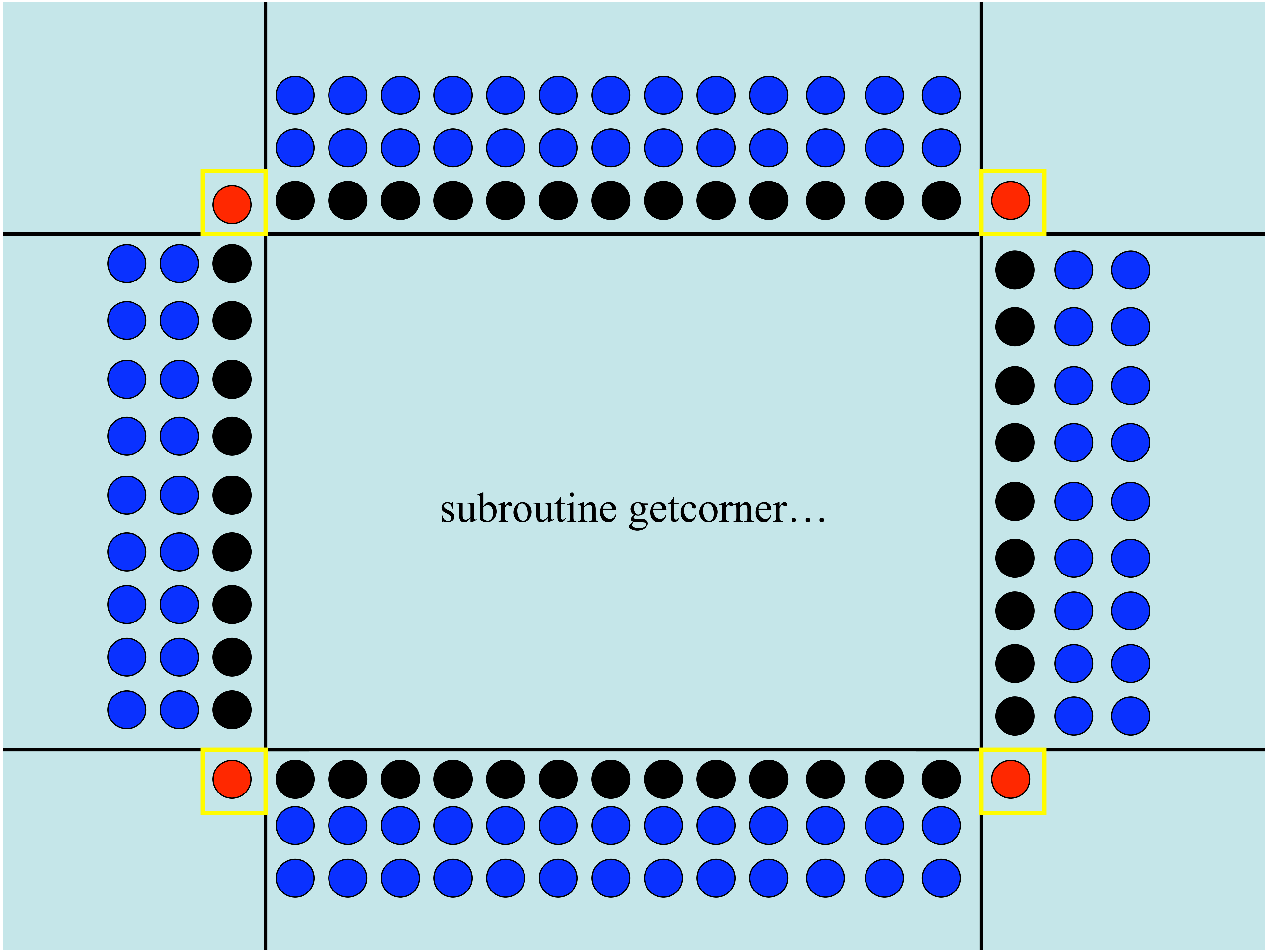


subdomain (tile)




subroutine comm_1…


subroutine comm_3…



subroutine getcorner…

MPI strategy

- Mostly non-blocking communications
  1. Call mpi_isend / mpi_irecv ….
  2. Go do some other work for awhile
  3. When data are needed … Call mpi_wait

(goal is to separate steps 1 and 3 as much as possible)

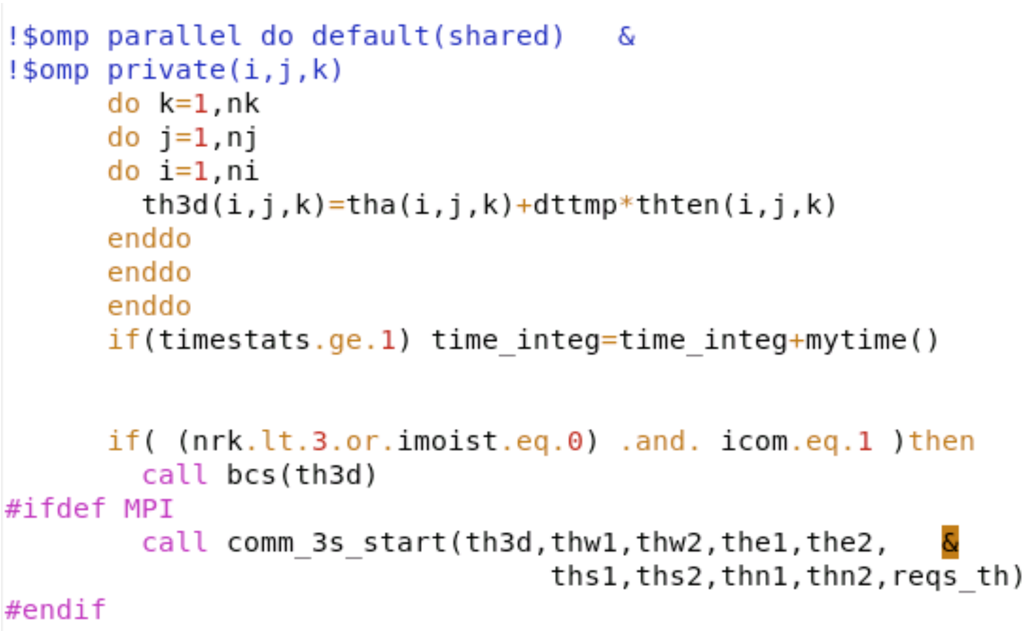
MPI strategy

- Mostly non-blocking communications

1. Call mpi_isend / mpi_irecv ….
2. Go do some other work for awhile
3. When data are needed … Call mpi_wait

(goal is to separate steps 1 and 3 as much as possible)

e.g., calculate new θ, start comm_3s ….



```
!$omp parallel do default(shared)   &
!$omp private(i,j,k)
      do k=1,nk
      do j=1,nj
      do i=1,ni
        th3d(i,j,k)=tha(i,j,k)+dttmp*thten(i,j,k)
      enddo
      enddo
      enddo
      if(timestats.ge.1) time_integ=time_integ+mytime()


      if( (nrk.lt.3.or.imoist.eq.0) .and. icom.eq.1 )then
        call bcs(th3d)
#ifdef MPI
        call comm_3s_start(th3d,thw1,thw2,the1,the2,   &
                                ths1,ths2,thn1,thn2,reqs_th)
#endif
```

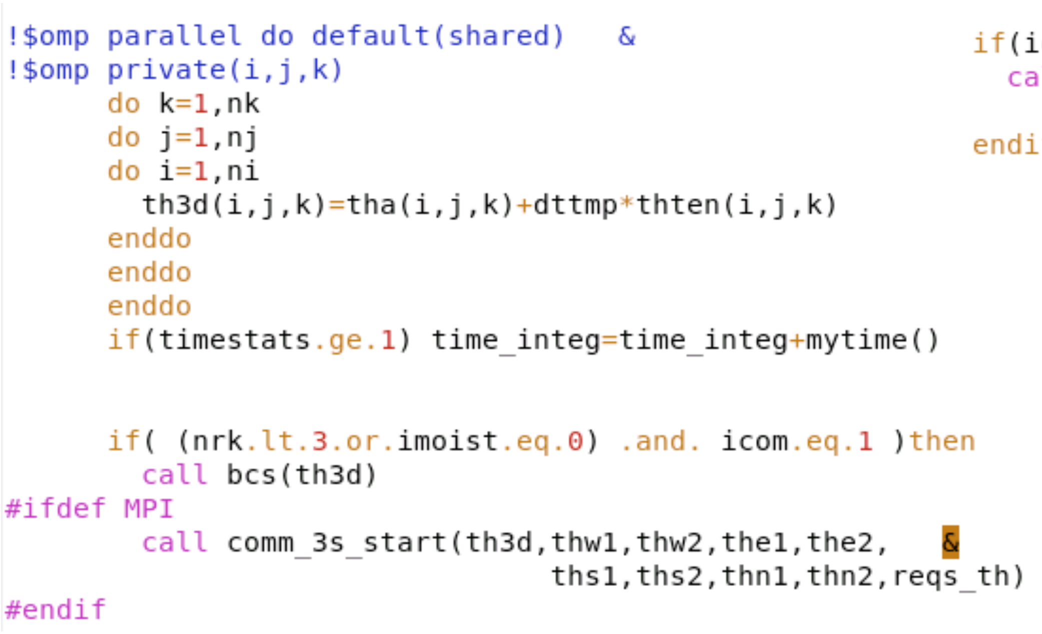
MPI strategy

- Mostly non-blocking communications

1. Call mpi_isend / mpi_irecv ….
2. Go do some other work for awhile
3. When data are needed … Call mpi_wait

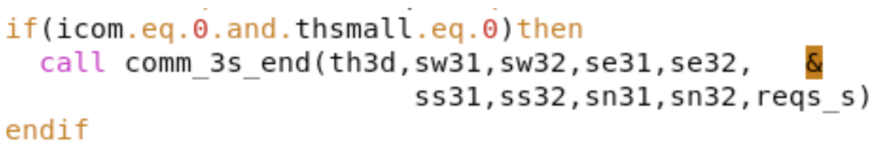
(goal is to separate steps 1 and 3 as much as possible)

e.g., calculate new θ, start comm_3s ….

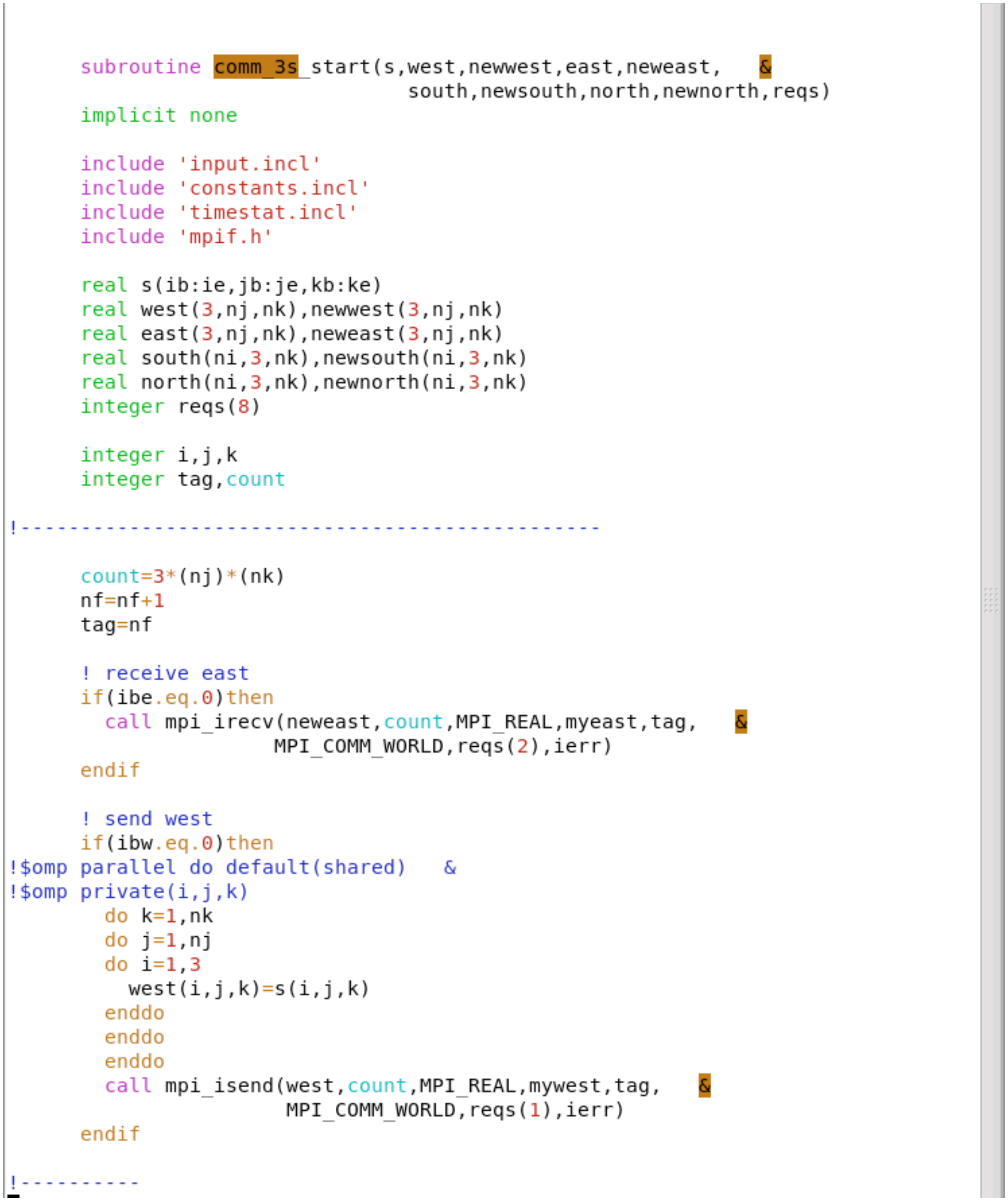


```
!$omp parallel do default(shared)   &
!$omp private(i,j,k)
      do k=1,nk
      do j=1,nj
      do i=1,ni
        th3d(i,j,k)=tha(i,j,k)+dttmp*thten(i,j,k)
      enddo
      enddo
      enddo
      if(timestats.ge.1) time_integ=time_integ+mytime()


      if( (nrk.lt.3.or.imoist.eq.0) .and. icom.eq.1 )then
        call bcs(th3d)
#ifdef MPI
        call comm_3s_start(th3d,thw1,thw2,the1,the2,   &
                                ths1,ths2,thn1,thn2,reqs_th)
#endif
```

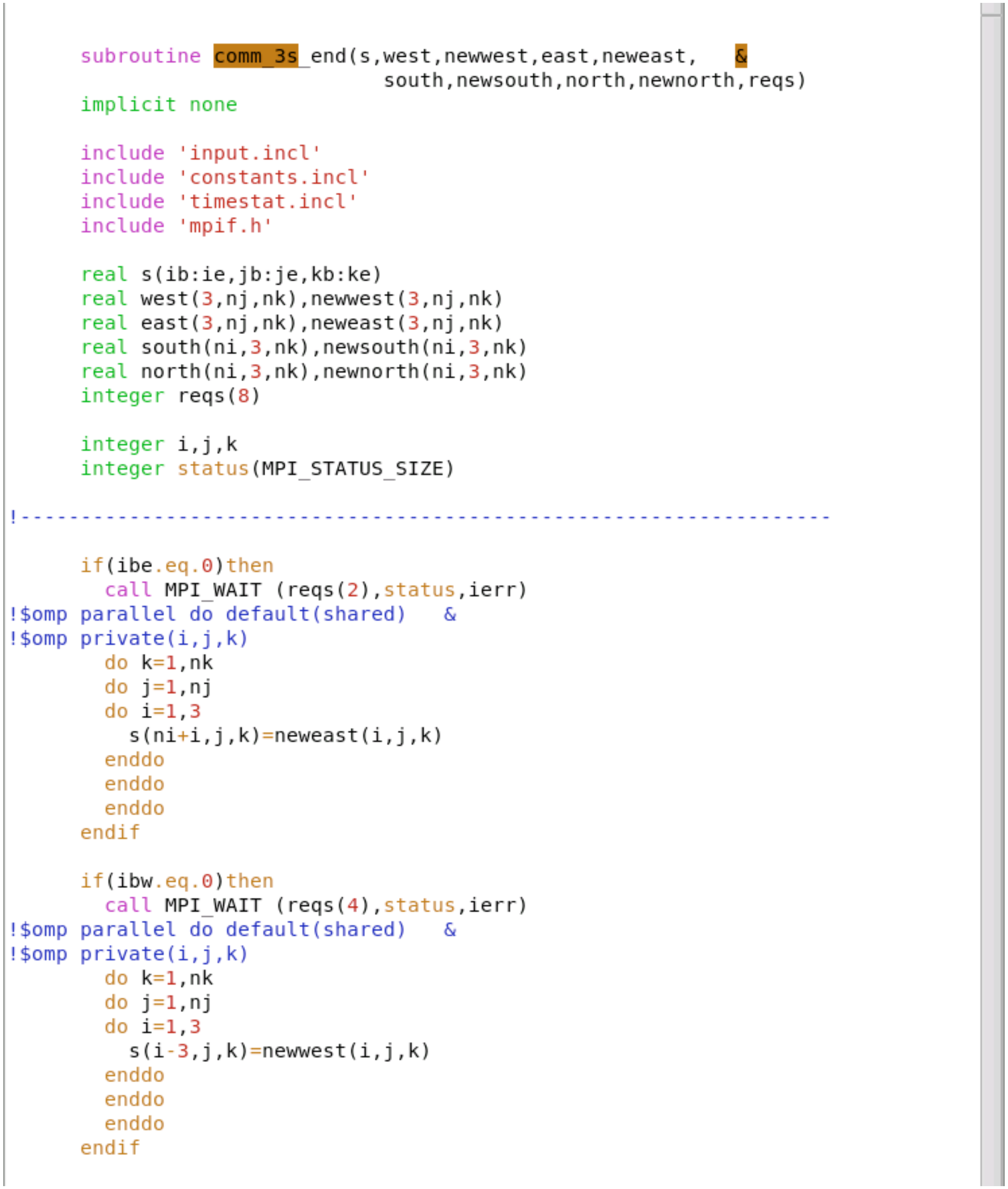
… do other calculations, then finish comm_3s



```
if(icom.eq.0.and.thsmall.eq.0)then
  call comm_3s_end(th3d,sw31,sw32,se31,se32,   &
                        ss31,ss32,sn31,sn32,reqs_s)
endif
```

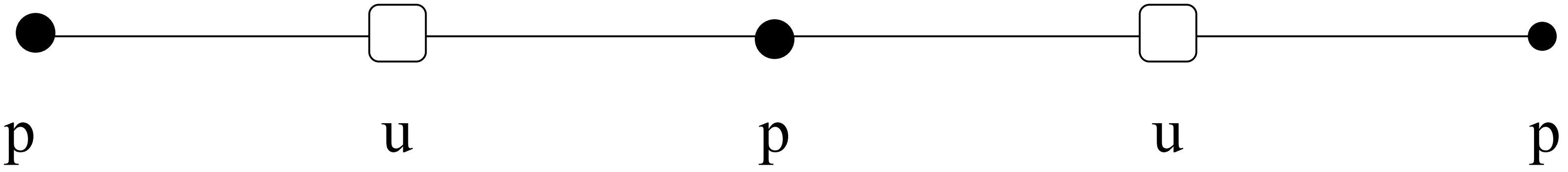


```
      subroutine comm_3s_start(s,west,newwest,east,neweast,   &
                                 south,newsouth,north,newnorth,reqs)
      implicit none

      include 'input.incl'
      include 'constants.incl'
      include 'timestat.incl'
      include 'mpif.h'

      real s(ib:ie,jb:je,kb:ke)
      real west(3,nj,nk),newwest(3,nj,nk)
      real east(3,nj,nk),neweast(3,nj,nk)
      real south(ni,3,nk),newsouth(ni,3,nk)
      real north(ni,3,nk),newnorth(ni,3,nk)
      integer reqs(8)

      integer i,j,k
      integer tag,count

!------------------------------------------------

      count=3*(nj)*(nk)
      nf=nf+1
      tag=nf

      ! receive east
      if(ibe.eq.0)then
        call mpi_irecv(neweast,count,MPI_REAL,myeast,tag,   &
                      MPI_COMM_WORLD,reqs(2),ierr)
      endif

      ! send west
      if(ibw.eq.0)then
!$omp parallel do default(shared)   &
!$omp private(i,j,k)
        do k=1,nk
        do j=1,nj
        do i=1,3
          west(i,j,k)=s(i,j,k)
        enddo
        enddo
        enddo
        call mpi_isend(west,count,MPI_REAL,mywest,tag,   &
                       MPI_COMM_WORLD,reqs(1),ierr)
      endif

!----------
```

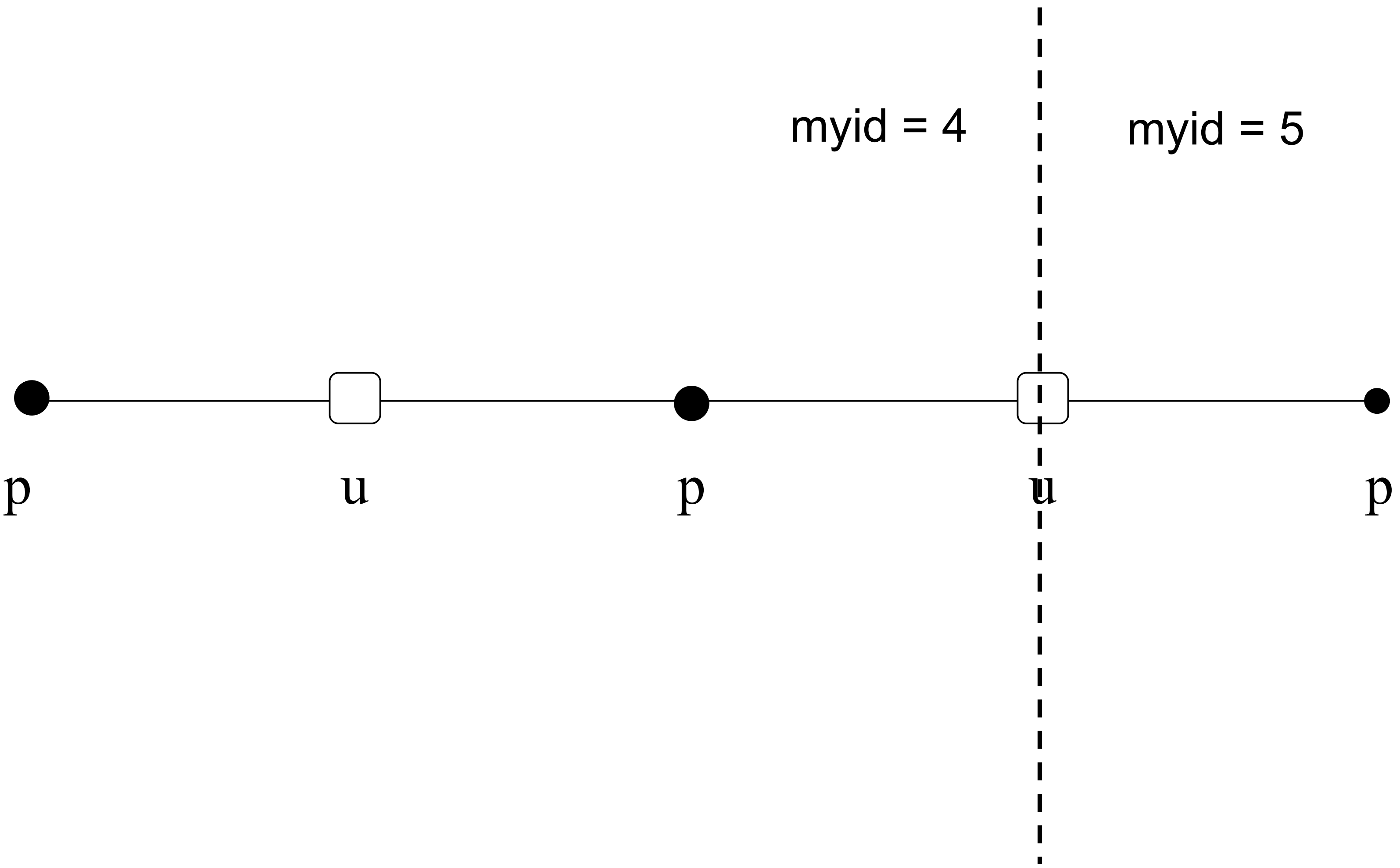


```
      subroutine comm_3s_end(s,west,newwest,east,neweast,   &
                              south,newsouth,north,newnorth,reqs)
      implicit none

      include 'input.incl'
      include 'constants.incl'
      include 'timestat.incl'
      include 'mpif.h'

      real s(ib:ie,jb:je,kb:ke)
      real west(3,nj,nk),newwest(3,nj,nk)
      real east(3,nj,nk),neweast(3,nj,nk)
      real south(ni,3,nk),newsouth(ni,3,nk)
      real north(ni,3,nk),newnorth(ni,3,nk)
      integer reqs(8)

      integer i,j,k
      integer status(MPI_STATUS_SIZE)

!--------------------------------------------------------------------

      if(ibe.eq.0)then
        call MPI_WAIT (reqs(2),status,ierr)
!$omp parallel do default(shared)   &
!$omp private(i,j,k)
        do k=1,nk
        do j=1,nj
        do i=1,3
          s(ni+i,j,k)=neweast(i,j,k)
        enddo
        enddo
        enddo
      endif

      if(ibw.eq.0)then
        call MPI_WAIT (reqs(4),status,ierr)
!$omp parallel do default(shared)   &
!$omp private(i,j,k)
        do k=1,nk
        do j=1,nj
        do i=1,3
          s(i-3,j,k)=newwest(i,j,k)
        enddo
        enddo
        enddo
      endif
```

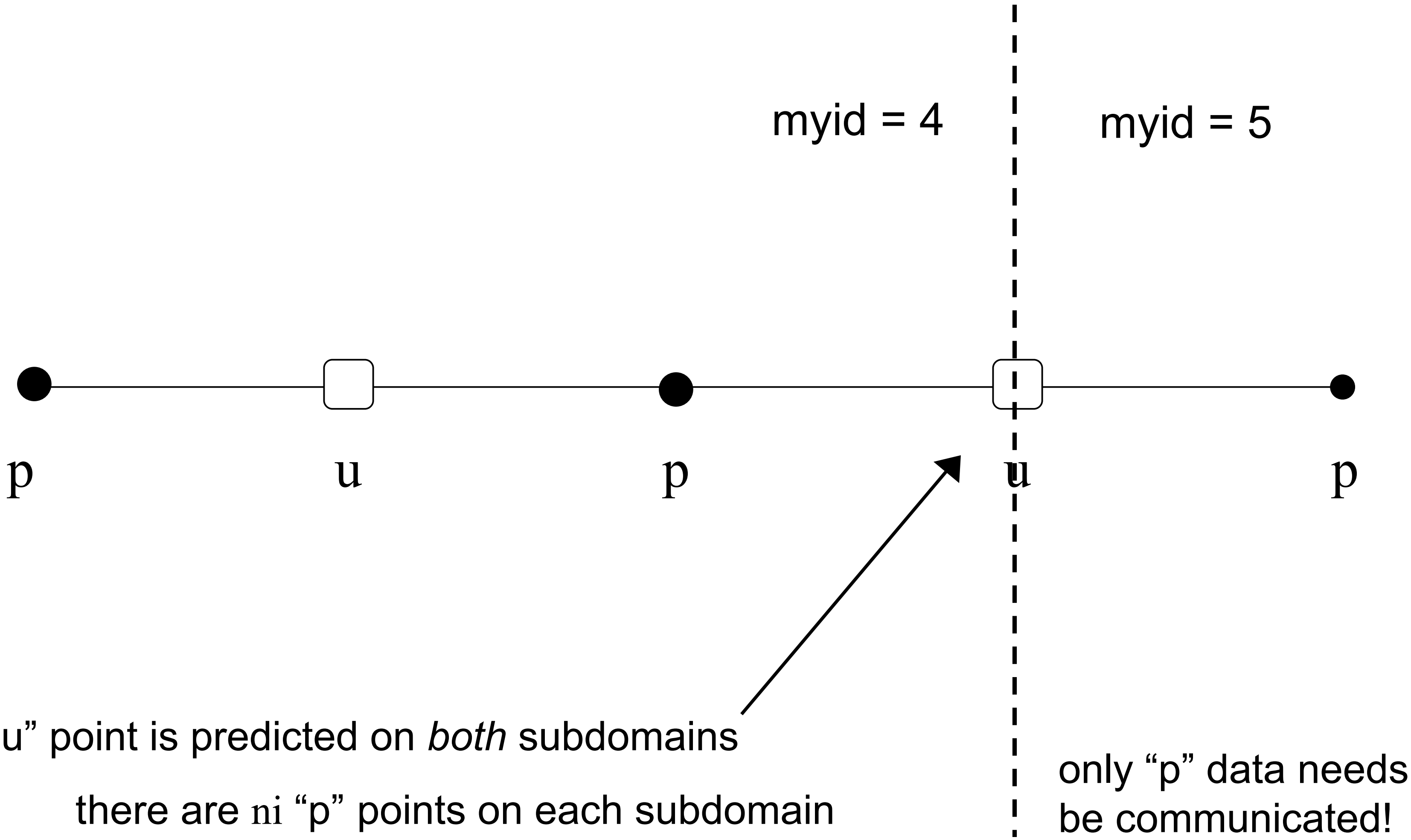
Communications on small (acoustic) steps



Communications on small (acoustic) steps



Communications on small (acoustic) steps



this “u” point is predicted on *both* subdomains

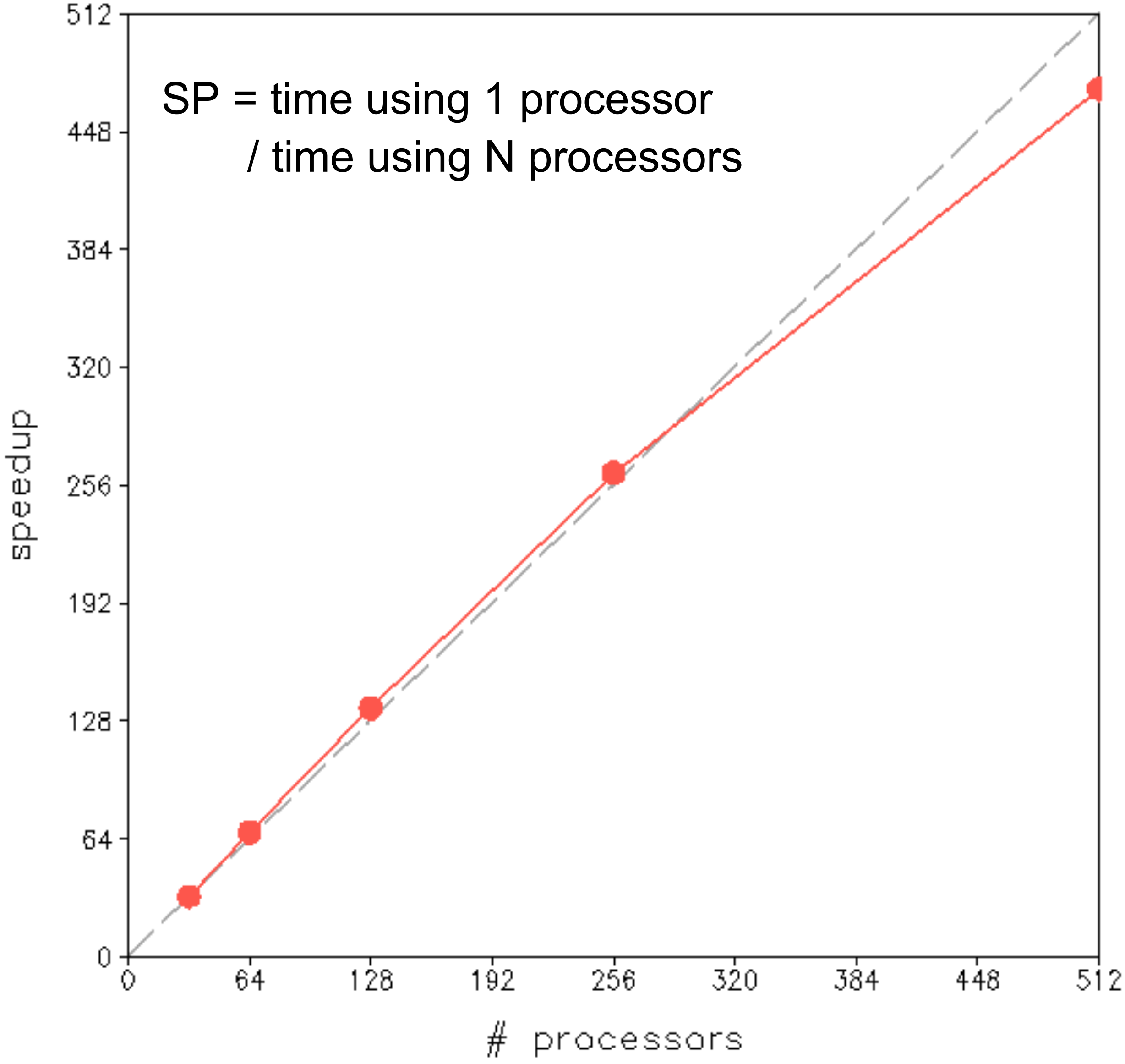
there are ni “p” points on each subdomain

there are ni+1 “u” points on each subdomain

only “p” data needs to be communicated!

NCAR’s bluefire: IBM Power 575, 4.7 GHz Power6 processors, infiniband switch, xlf compiler

3d hurricane simulation, 480 × 480 × 100 grid points, 3,600 time steps

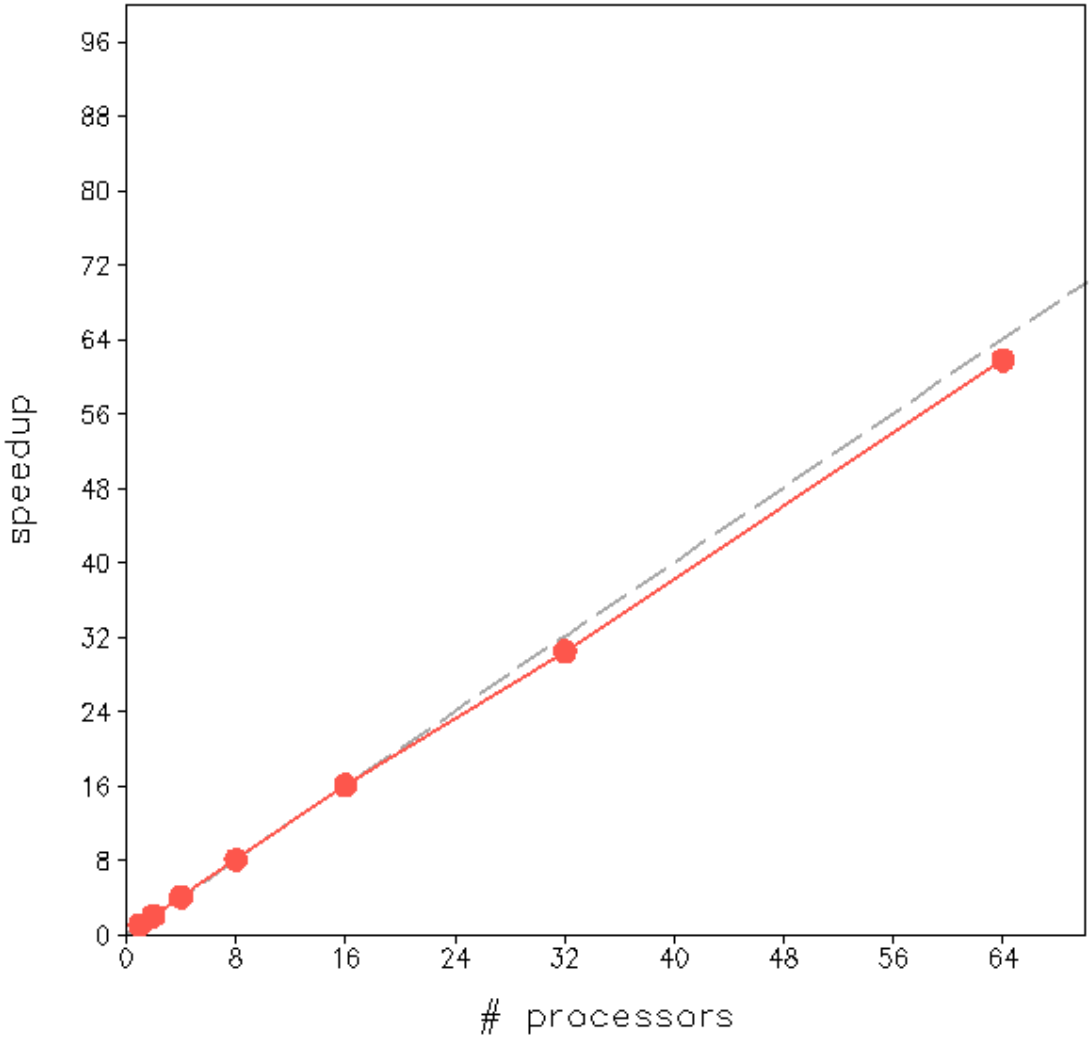


SP = time using 1 processor
/ time using N processors

NCAR's bluefire: 1-64 processors

(bluefire has 32 processors per node)

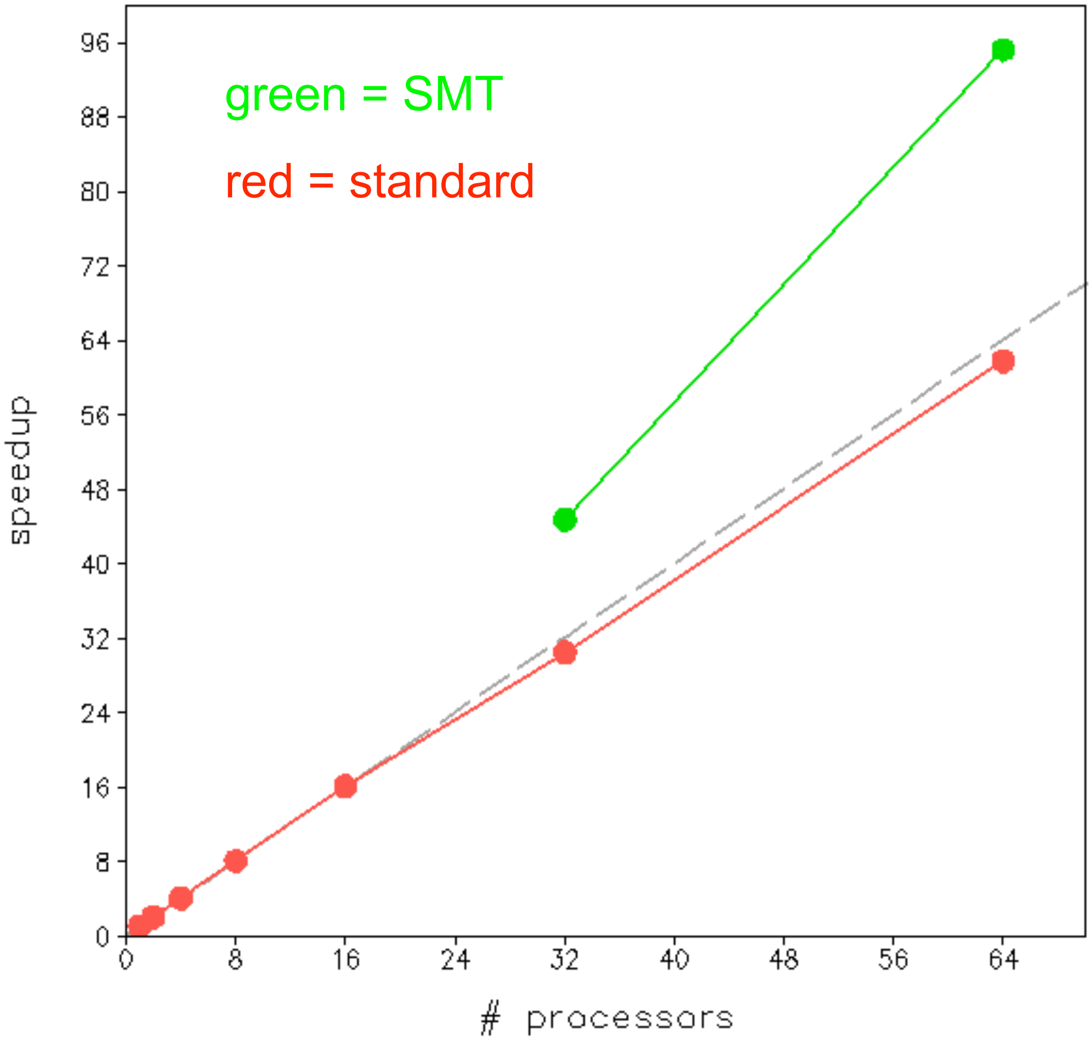
3d hurricane simulation, 480 × 480 × 50 grid points, 600 time steps



<u>NCAR’s bluefire</u>: 1-64 processors

(bluefire has 32 processors per node)

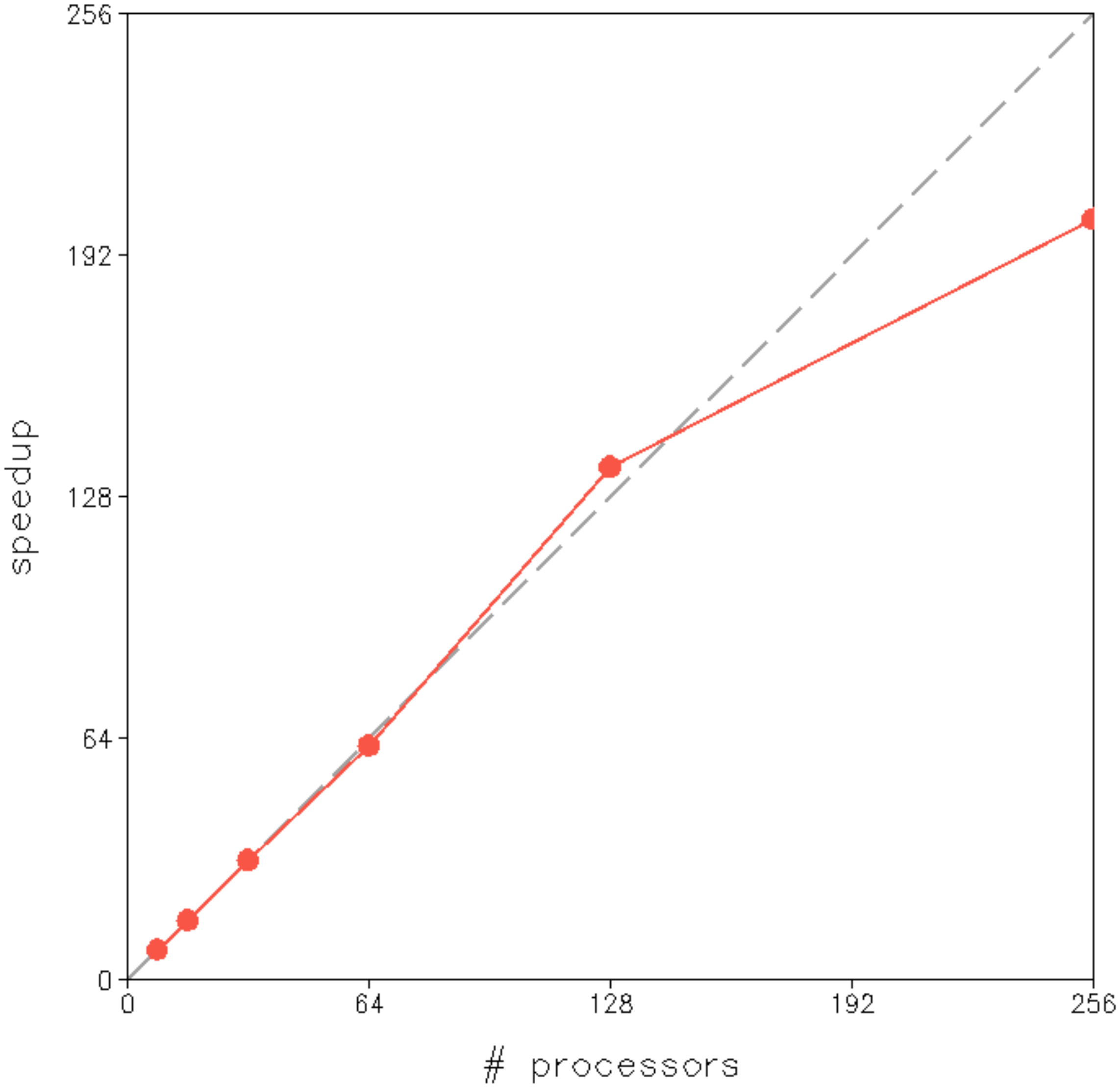
3d hurricane simulation, 480 × 480 × 50 grid points, 600 time steps



SHARCNET's saw: 2,688 processors, InfiniBand interconnect

8 processors per node, Intel Xeon 2.83 GHz, Intel fortran compiler

3d hurricane simulation, 480 × 480 × 100 grid points, 600 time steps


speedup
# processors