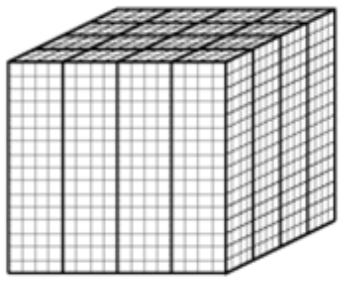
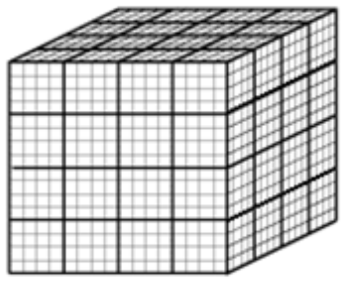
New version of EULAG with 3d domain decomposition

Zbigniew Piotrowski, Institute of Meteorology and Water Management, Warsaw, Poland

Codename Eulag 3p


2D


3D

Description of new features and optimizations

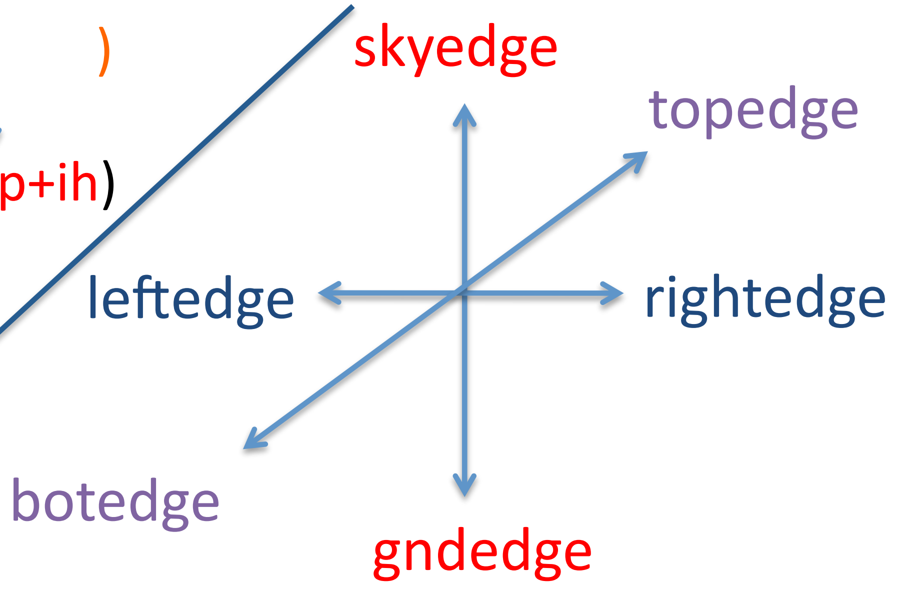
Loops and boundaries

Modified array declarations: 1-ih:mp+ih, l)

dimension u(1-ih:np+ih, 1-ih:mp+ih, 1-ih:lp+ih)

Loops throughout the code are now:

```
 do k=1,lp  (instead of do k=1,l )
      do j=1,mp
          do i=1,np
              ……
           enddo
      enddo
  enddo
```



New boundaries: gndedge and skyedge

```
If(ibcz.eq.1) then
       if (gndedge.eq.1) then
        do  j=jllim,julim
         do  i=illim,np
        v1(i,j,1)=vdyf(x(i-1,j,1),x(i,j,1),f1(i,j,1),
    *              .5*(h(i-1,j,1)+h(i,j,1)))
          enddo
        enddo
     endif  ! gndedge = 1
```

Processor grid and configuration

Processor grid is now defined by nprocz as well, e.g.:

```
        parameter (nprocx=4, nprocy=2, nprocz=4)
```

Processor position on the grid is now defined by lpos as well.
E.g., retrieval of the actual position (ia,ja,ka) from (i,j,k)
on a given processor is defined by:

```
 do 4 k=1,lp
     ka = (lpos-1)*lp + k
        do 4 j=1,mp
           ja = (mpos-1)*mp + j
              do 4 i=1,np
                 ia = (npos-1)*np + i
```

Updates and global operations

Update calls have now additional arguments, e.g.:

```
 call update(  ue,np,mp,lp,np,mp,lp,iup)
```

also there is a new type of update in the vertical direction only:

```
call updategs(pe,np+rightedge,mp+topedge,lp+skyedge,
     .                  np+1,mp+1,lp+1,iupz)
```

akin to iupx,iupy

Global operations (max,min,sum) also need additional arguments, e.g.:

```
 dftav=globsum(temp,1-ih,np+ih,1-ih,mp+ih,1-ih,lp+ih,
     .              1,np,1,mp,1,lp)
```

Optimizations

Petascale simulations with very large number of cores demand more focus on optimizing communication layer.

Processor geometry setup - new default Cartesian MPI topology option parameter(icart=1)

- Using icart=1 informs MPI system about the geometry of the task
- Results in several % smaller wall time, allows for more effective profiling in SCALASCA which is now able to understand mesh structure
- Invisible to the rest of the Eulag code …
- …. but, processors are usually numbered differently than for traditional icart=0 option, if you need to know how neighboring cores are distributed,

use peGNW,peGSW,peGSE,peGNE,
. peGW,peGE,peGN,peGS,
. peZNW,peZSW,peZSE,peZNE,
. peZW,peZE,peZN,peZS,
. peZ,peG,peW,peE,peN,peS,
. peNW,peSW,peSE,peNE

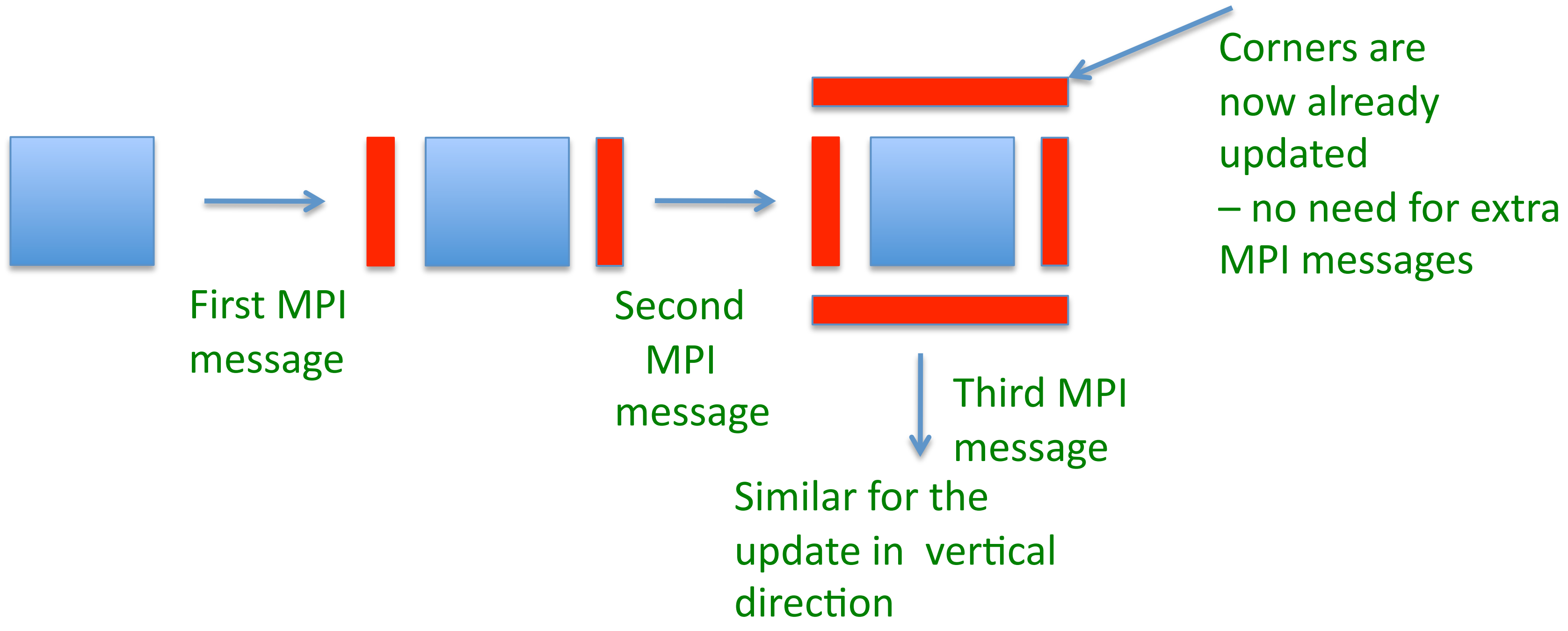
Z stands for “Zenith”

G stands for “Ground”

(replaces traditional peleft,peright,peabove,pebelow, etc.)

Brand new update subroutine

• Traditional set of update subroutine (update, updatelr, updatebt, updatew, update2) replaced with one subroutine update3dsplitn
• Invisible to the users, traditional calls to update, updatelr, updatebt, etc. are now WRAPPERS only (translating simple old way of calling to the fully universal update3dsplitn)
• New update3dsplitn limits number of MPI messages to three, but two of them communicating the domain WITH HALO


Corners are now already updated – no need for extra MPI messages
First MPI message
Second MPI message
Third MPI message
Similar for the update in vertical direction

Further halo update optimizations

- New update3dsplitn allows for overlapping computations and communications
- New set of OPTIONAL subroutines are defined and can be used as in the following example:

```
    call updatelrbeg(f1,np+rightedge,mp,lp,np+1,mp,lp,1,1)
     do 403 j=1,lp+skyedge
     do 403 i=1,np
 403 f2(i,1,j)=donor(c2,c2,v2(i,1,j))
     call updatelrend(f1,np+rightedge,mp,lp,np+1,mp,lp,1,1)
```

Last parameter for update..beg and update…end is a number of buffer (because one may wish to update more than one variable)

- Full update can be done in a sequence:

```
call updatelrbegf(f1,np+rightedge,mp,lp,np+1,mp,lp,1,1)
…..
call updatelrendf(f1,np+rightedge,mp,lp,np+1,mp,lp,1,1)
call updatebtbegf(f1,np+rightedge,mp,lp,np+1,mp,lp,1,1)
…..
call updatebtendf(f1,np+rightedge,mp,lp,np+1,mp,lp,1,1)
call updategsbegf(f1,np+rightedge,mp,lp,np+1,mp,lp,1,1)
……
call updategsendf(f1,np+rightedge,mp,lp,np+1,mp,lp,1,1)
```

Further halo update optimizations #2

- For nonperiodic boundary conditions, you don’t need to update outer boundaries of the computational domain, which are usually far from each other inside the machine.
The only exception is *vbcad* subroutine.

- Therefore, there is an important change in DEFAULT behavior –
For nonperiodic b.c. there is no update on outer domain borders

- On the other hand, for periodic boundary conditions it is often unnecessary to update variables inside the domain. For these two purposes, new set of BORDER ONLY updates are defined:

```
if(ibcz.eq.1) then
     call updategsbor(p,np,mp,lp,np,mp,lp,1) ! updategsborbeg and updategsborend possible
      if(skyedge.eq.1) then
        do 10 j=1,mp
          do 10 i=1,np
 10           p(i,j,lp)=p(i,j,lp+1)
       endif
```

Global communication optimizations

• Global summation, max and min operations are necessary for elliptic solver but EXTREMELY expensive (and relative cost growing)

•Therefore, there is a need to minimize number of AlltoAll operations

•It is done by global operations (sum, max, min or combined) on vectors

Example:

globsumvbor - sums u,v,w outflow, inflow and weights in vbcad within 1 global communication instead of 9 (especially useful when you call vbcad every timestep)

globsumv, globmaxv, globsumaxv – sums, finds max, or both at the same global communication for vector of data

Example (from gcrk):

```
eetabs(1)=qrlsum
eetabs(2)=qrlmax
call globsumaxv(eetabs,eetabd,2)
  eer=eetabd(1)
eem=amax1(eem,sqrt(eetabd(2)))
```

```
call MPI_OP_CREATE
.(sumax,commute,MPI_SUMAX,err)
 function sumax(wrkin,wrkinout,ihalf,itype)
      do i=1,ihalf
      wrkinout(i)=wrkin(i)+wrkinout(i)
      wrkinout(ihalf+i)=amax1(wrkinout(ihalf+i)
.                             ,wrkin(ihalf+i))
```

Other optimizations

• In principle, it is not necessary to compute all the norms for exit conditions from GCRK iterations at each timestep since they need expensive global communication. Threshold for these computations:

```
parameter(itertrem=0.5)  ! Could be any number between (0,1)
```

defines at which fraction of the last number of GCRK iterations, the norms for GCRK exit conditions will be computed.

• Computations of precon_bcz coefficients now takes place once per gcrk call in precon_bcz_ini

• Parallel tridiagonal solver, necessary if nprocz > 1 is implemented in tdmapar subroutine. Variables isequp, iseqdn initialized in blockdata blanelas choose details of algorithms (default: 7, failsafe: 1)

• Parallel NETCDF long and short tape write is now called by iowrite/ioread and iowrsh/iorsh subroutines as the standard Fortran tape

Compiler setup

• Cray supercomputers and Linux clusters, very popular these days, offer various compilers (PGI, Intel, Cray, Pathscale, GNU), which makes maintenance of library, compiling and submission scripts difficult

• Set of new environmental variables has been introduced to facilitate switching between the compilers, optimization levels and submission methods:

```
##################################################################################
##
#### Choose compiler if more then one available
##
##   0 - Default compiler
##   1 - PGI compiler
##   2 - Intel compiler
##   3 - Pathscale compiler
##   4 - Cray  compiler
##   5 - GNU compiler
##
###################################################################################
setenv COMPILER 4
```

```
###################################################################
###########
####  Define level of compiler optimization
##    0  - no optimization at all (for very fast compilation)
##    1  - full set of diagnostic and debugging options
##    2  - default optimization   (different for different machines)
##    3  - strong  optimization   (recommended by compiler vendors)
##    4  - maximum optimization   (usually -O4 or -O5 with IPA)
###################################################################
###########
setenv COMPOPTI 0
###################################################################
###########
##
#### Define special option for code instrumentation for profiling
#### Default: setenv CODEINST 0
##   0 - No instrumentation (default)
##   1 - Scalasca instrumentation with skin
##   2 - Tau compiler instrumentation
##   3 - Tau PDT instrumentation
##
###################################################################
###########
setenv CODEINST 0
```

Architectures

Recently, Eulag has been successfully run on the following modern architectures (setups available):

- Intel Linux cluster (JANUS @ CU, Boulder)
- IBM Power 7 (ICM, Poland)
- Bluegene/P (ICM, Poland)
- Cray XE 6 (CSCS, Switzerland)
- Cray XT5m (NCAR, Boulder)

Optimizations discussed are not fully tested, IFASTSUBS 0 and IOVER 0 available to switch to generic "failsafe" mode without optimizations

Known problems:

- Each new machine is likely to cause some problems
- Standard optimizations of some newer versions of PGI

may not work, always try COMPOPTI 0 to see if it helps.

Parallelization status

- Tested and work: 2D/3D Eulerian/SL solver, GCRK, bulk moist models, ILES/SGS 1/SGS 2, ANALIZ 1 mode, velocity predictor

- Untested: WORD 4 time integration, grid adaptivity, , SGS 2 for ibcz=1 seems incomplete, although parallelized

- Not parallelized/implemented yet: single core mode, zonal absorbers, parallel diagnostic packages, efficient precon_bcz for ibcz=1, MHD, unstructured

- Serial NETCDF and slices subroutines are not fully parallelized and/or not up to date