serialization

Serialization (1/4)

- Effort to synchronize Fortran with C++
- Unit-test Fortran sections against C++ implementation
- Serialization = write input / output of section to file

- **Problem:** Additional code…
  - #ifdef
  - can be significant
  - can confuse developer

- **Proposal:** Custom pre-processor
  - serialization directives !$ser
  - significant reduction of additional code
  - appear as comments to developer

Serialization (2/4)

- Example: Coriolis

```
#ifdef _SERIALIZE
call fs_SetSavePointName('CoriolisUnittest.Apply-in')
call fs_AddSavePointInfo('LargeTimeStep', ntstep)
call fs_AddSavePointInfo('RKStageNumber', 0)
call fs_AddSavePointInfo('SmallTimeStep', 0)
call fs_WriteData('u_nnow', u(:,:,:,nnow))
call fs_WriteData('v_nnow', v(:,:,:,nnow))
call fs_WriteData('u_tens', utens)
call fs_WriteData('v_tens', vtens)
#endif

  DO k = 1 , ke
    DO j = jstartu, jendu
      DO i = istartu, iendu
        z_fv_north = fc(i,j)   * ( v(i,j  ,k,nn) + v(i+1,j  ,k,nn) )
        z_fv_south = fc(i,j-1) * ( v(i,j-1,k,nn) + v(i+1,j-1,k,nn) )

        ...

      ENDDO
    ENDDO
  END DO

#ifdef _SERIALIZE
call fs_SetSavePointName('CoriolisUnittest.Apply-out')
call fs_AddSavePointInfo('LargeTimeStep', ntstep)
call fs_AddSavePointInfo('RKStageNumber', 0)
call fs_AddSavePointInfo('SmallTimeStep', 0)
call fs_WriteData('u_tens', utens)
call fs_WriteData('v_tens', vtens)
#endif
```

Serialization (3/4)

- Example: Coriolis

```
!$ser savepoint CoriolisUnittest.Apply-in LargeTimeStep=ntstep RKStageNumber=0 SmallTimeStep=0
!$ser data u_nnow=u(:,:,:,nnow) v_nnow=v(:,:,:,nnow) u_tens=utens v_tens=vtens

  DO k = 1 , ke
    DO j = jstartu, jendu
      DO i = istartu, iendu
        z_fv_north = fc(i,j)   * ( v(i,j  ,k,nn) + v(i+1,j  ,k,nn) )
        z_fv_south = fc(i,j-1) * ( v(i,j-1,k,nn) + v(i+1,j-1,k,nn) )

        ...

      ENDDO
    ENDDO
  END DO

!$ser savepoint CoriolisUnittest.Apply-out LargeTimeStep=ntstep RKStageNumber=0 SmallTimeStep=0
!$ser data u_tens=utens v_tens=vtens
```

- Lightweight Python script (< 700 lines)
- Small set of powerful directives

```
!$ser init          !$ser registertracer !$ser zero
!$ser option        !$ser tracer         !$ser perturb
!$ser register      !$ser savepoint      !$ser cleanup
!$ser data          !$ser verbatim
```

Serialization (4/4)

- Show some other code examples
- Total of 450 lines of directives
  (mostly in lmorg.f90 and dynamical core)

Proposal for v5.2

- Integrate serialization directives
- Deliver official version with Python script and a Makefile target to build serialization

Discussion

- Who will maintain serialization?

halo-update

New halo-update API (1/4)

- API of GCL and exchg_boundaries() are not compatible
  - initialization vs. on-the-fly
  - static vs. dynamic fields
  - asynchronous vs. synchronous
  - CPU/GPU vs. CPU-only
- Idea was to redesign an API which works for both Fortran and GCL
- Design proposal exists, but currently no resources for implementation

New halo-update API (2/4)

```
kzdims(1:24)=(/ke,ke,ke1,ke,ke,ke,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0/)
CALL exchg_boundaries                                                  &
 (50+nnew, sendbuf, isendbuflen, imp_reals, icomm_cart, num_compute,   &
  ie, je, kzdims, jstartpar, jendpar,                                  &
  nbl_exchg, nboundlines, my_cart_neigh,                               &
  lperi_x, lperi_y, l2dim,                                             &
  17000+nexch_tag, ldatatypes, ncomm_type, izerror, yzerrmsg,          &
  u(:,:,:,nnew), v(:,:,:,nnew), w(:,:,:,nnew), t(:,:,:,nnew),          &
  pp(:,:,:,nnew), qrs(:,:,:) )

IF ( lzconv ) THEN
  IF ( lprog_tke ) THEN
    IF (itype_turb /= 3 .OR. ntke == 0) THEN
      zntke = nnew
    ELSE
      zntke = ntke
    END IF
    kzdims(1:24)=(/ke,1,ke1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0/)
    CALL exchg_boundaries                                                &
     (0, sendbuf, isendbuflen, imp_reals, icomm_cart, num_compute,       &
      ie, je, kzdims, jstartpar, jendpar,                                &
      nbl_exchg, nboundlines, my_cart_neigh,                             &
      lperi_x, lperi_y, l2dim,                                           &
      18000+nexch_tag, .FALSE., ncomm_type, izerror, yzerrmsg,           &
      dqvdt(:,:,:), qvsflx(:,:), tke(:,:,:,zntke) )
  ELSE
    kzdims(1:24)=(/ke,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0/)
    CALL exchg_boundaries                                                &
     (0, sendbuf, isendbuflen, imp_reals, icomm_cart, num_compute,       &
      ie, je, kzdims, jstartpar, jendpar,                                &
      nbl_exchg, nboundlines, my_cart_neigh,                             &
      lperi_x, lperi_y, l2dim,                                           &
      18000+nexch_tag, .FALSE., ncomm_type, izerror, yzerrmsg,           &
      dqvdt(:,:,:), qvsflx(:,:) )
  END IF
ELSE
```

New halo-update API (3/4)

```
INTEGER, SAVE :: iexchg = -1

IF (iexchg < 0) THEN
  iexchg = gcl_CreateHaloExchange( "MainHaloUpdateRK" )
  CALL gcl_AddFieldToHaloExchange( iexchg, nbl_exchg, jstartpar, jendpar, u(:,:,:,nnew) )
  CALL gcl_AddFieldToHaloExchange( iexchg, nbl_exchg, jstartpar, jendpar, v(:,:,:,nnew) )
  CALL gcl_AddFieldToHaloExchange( iexchg, nbl_exchg, jstartpar, jendpar, w(:,:,:,nnew) )
  CALL gcl_AddFieldToHaloExchange( iexchg, nbl_exchg, jstartpar, jendpar, t(:,:,:,nnew) )
  CALL gcl_AddFieldToHaloExchange( iexchg, nbl_exchg, jstartpar, jendpar, pp(:,:,:,nnew) )
  CALL gcl_AddFieldToHaloExchange( iexchg, nbl_exchg, jstartpar, jendpar, qrs(:,:,:) )
  IF (lconv) THEN
    CALL gcl_AddFieldToHaloExchange( iexchg, nbl_exchg, jstartpar, jendpar, dqvdt(:,:,:) )
    CALL gcl_RegisterField( qvsflx, "qvsflx" )
    CALL gcl_AddNonDycoreFieldToHaloExchange( iexchg, nbl_exchg, jstartpar, jendpar, qvsflx(:,:) )
  ENDIF
  IF (lprog_tke) THEN
    IF (itype_turb /= 3 .OR. ntke == 0) THEN
      zntke = nnew
    ELSE
      zntke = ntke
    END IF
    CALL gcl_AddFieldToHaloExchange( iexchg, nbl_exchg, jstartpar, jendpar, tke(:,:,:,zntke) )
  ENDIF
  DO iztrcr = 1, trcr_get_ntrcr()
    CALL trcr_get(izerror, iztrcr, ptr_tlev=nnew, ptr=ztrcr)
    IF (izerror /= 0_iintegers) THEN
      yzerrmsg = trcr_errorstr(izerror)
      CALL model_abort(my_cart_id, izerror, yzerrmsg, yzroutine)
    ENDIF
    CALL gcl_AddFieldToHaloExchange( iexchg, nbl_exchg, jstartpar, jendpar, ztrcr(:,:,:) )
  ENDDO
ENDIF
CALL gcl_DoExchange( iexchg )
```

New halo-update API (4/4)

Proposal for v5.2

- Use #ifdef GCL_COMM to include both halo-exchanges in code
- Replace only bare minimum of halo-updates for now (i.e. within timeloop, GPU fields)
- Use STOP for unimplemented one's

BCs

Boundary conditions (1/1)

- New src_lbc.f90 module (→ see talk from this morning)
- Used in several modules (lmorg.f90, dycore)
- **Guiding principle** Group boundary conditions with halo-updates as much as possible

Proposal for v5.2

- Introduce src_lbc.f90
- Replace BCs at as many places as possible
- Fix BC bugs in Fortran dynamical core

single
precision

Single precision (1/1)

- Accepted into v5.1 and sent out for testing
- **But…**
  - assimilation does not work

```
IF (     (ntstep < 11)                                                  &
    .OR. (MOD(ntstep,20) < INT( rmod(tconbox,dt,epsy)-epsy ))) THEN
```

  - many options not tested (e.g. lseaice = .true.)

Proposal

- New POMPA task for SP in assimilation
- Make developers aware of SP issues
- Add SP to technical testsuite
- Modify COSMO Coding Standards

dycore init

Dynamics initializations (1/1)

- org_runge_kutta() contains a section of initializations which are done if (ntstep == ntstart)
- Example: Computation of t0 and dt0dz (as a function of itheta_adv)
- COSMO coding standards

> A package / component shall provide different set-up and running procedures (if necessary), each with a single entry point. All initialization of static data must be done in the set-up procedure and these data must not be altered by the running procedures.

Proposal for v5.2

- Move into init_dynamics()

assml / relaxation

Assml / relaxation (1/1)

- Relaxation is implemented in C++ dynamical core
- Worflow in Fortran part is…
  1. dynamical core
  2. assimilation
  3. relaxation + time-filtering (LF)
- Relaxation also acts in boundary zone

Proposal for v5.2

- Split relaxation from time-filtering
- Move relaxation to after dynamical core
- Apply relaxation only in compute domain
- Ensure correct BC in dynamics before relaxation