Fieldextra

Jean-Marie Bettems / MeteoSwiss
13.09.2012
Lugano (CH)

Identity card (1)

- Generic tool to process model data and gridded observations
  - implement a set of primitive operations, which can be freely combined and iterated (**toolbox**)
  - single **Fortran program** controlled by **namelists**

- File based input/output ...
  - support both **GRIB1** and **GRIB2** (input/output)
  - support **local extension** of GRIB standard
  - understand **naming conventions** of COSMO files
  - rich set of output format in addition to GRIB (NetCDF, CSV, XML ...)

- Primary focus is the production environment
  - **high quality standard** (design, implementation, exceptions, testing)
  - **optimized** code (io, memory, cpu and elapsed time)
  - comprehensive **diagnostic** and **profiling**
  - inter-process communication (support parallel production suite)

Identity card (2)

- About 90k lines of **Fortran 2003**
  - +10k lines yearly, +20k lines last year
  - Linked with **DWD grib library** (GRIB1), **ECMWF grib API** (GRIB2), **JasPer** (JPEG in GRIB2), **NetCDF** library (NetCDF), **hdf5** library (for NetCDF), **zlib** library (for NetCDF) and some **COSMO modules**
  - **OpenMP** implementation for shared memory parallelism
  - **Standalone package** available on COSMO web site, including source code for all above mentioned libraries http://www.cosmo-model.org/content/support/software/default.htm

Identity card (3)

- **Portable** code
  - Test platforms: **Cray Opteron, IBM Power**
  - Test compilers: **GNU, Intel, IBM**
    (IBM for OpenMP code still a work in progress)
  - Should work on any UNIX / Linux / Mac platform
- **Documented** code
  - User manual, examples, FAQ, developer manual …
- **Community support**
  - cosmo-fieldextra@cosmo-model.org
- **Limitations**
  - Complex namelists, steep learning curve

Usage

- **COSMO software** (licensed)
- **COSMO adaptor** for the EUMETNET programme SRNWP interoperability

- Core **non-graphical NWP production tool** at MeteoSwiss
  - About **15’000 products** per day generated with fieldextra, representing more than **200 GB data**
  - Products derived from **COSMO-2**, **COSMO-7**, **COSMO-LEPS**, **PEPS**, **IFS**
  - Thresholds and regions based **warnings** for the ‘Common Information Platform for Natural Hazards’, developed for the Swiss government
- **COSMO-LEPS production** at ECMWF
- **FABEC production** at DWD
  - Additional products for the German flight control
- **Others**
  - NMA, RHM, …

Activities since last COSMO GM

- **COSMO GM 2011 : release 10.4.0**
  COSMO GM 2012 : release 10.5.3 (private release)

- Bug correction, internal code improvements
- Support input files mixing GRIB1 and GRIB 2 records
- Consolidate **GRIB2** support
- Implement **NetCDF** output
- Code **optimization**
- Implement **shared memory parallelism** (OpenMP)
- Implement **MOS** corrections
- Implement **EPS** based standard deviation and quantiles difference
- And many more refinements …

- New **FAQ** ('Frequently asked questions')
- **Tutorial** at ARPA-SIMC
  Tutorial for 'Capacity Building' event

Shared memory parallelism

- Shared memory multitasking is available and implemented with **OpenMP directives**

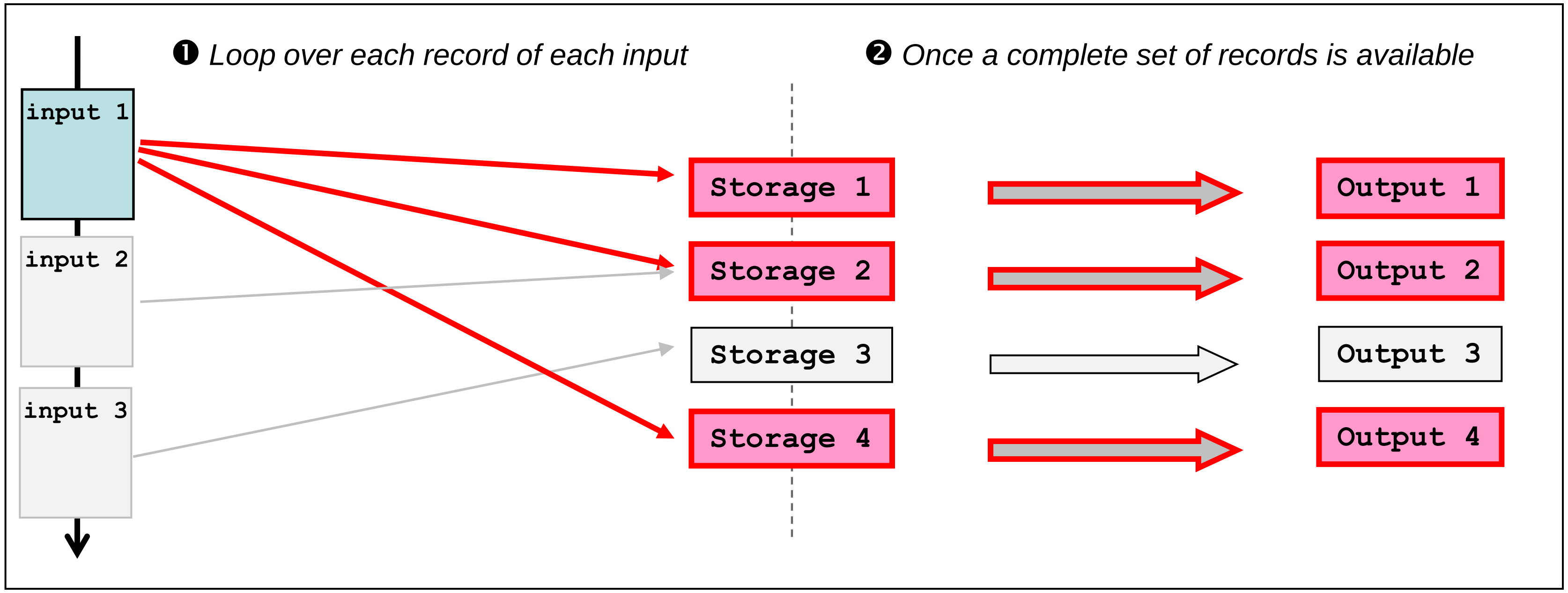
- **Two levels of parallelism** are implemented and can be simultaneously used
  - parallel production of **output** (outer loop parallelism)
  - parallelization of some of the **algorithms** used during the production of each output (inner loop parallelism)

- Two (exclusive) types of **algorithm parallelization** are available
  - Parallel computation when the same operator (e.g. regridding) is applied on many records within the current iteration
  - Grid points partitioning (computation of derived field only)

- No distributed memory parallelism
- No parallelization of input processing

Shared memory parallelism

Parallel production of output (outer loop parallelism, marked with ——— below)



For each output the following operations are applied in parallel:

(1) For each record in turn :

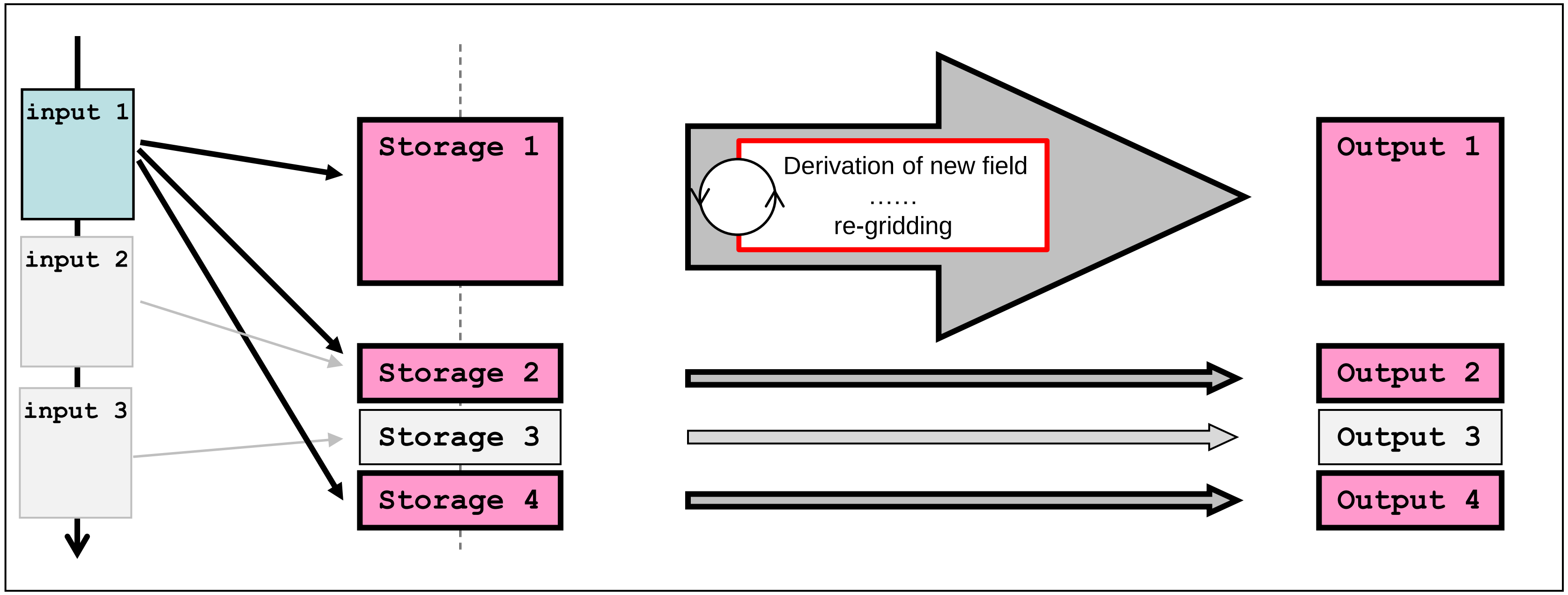
check use of current record , process and store record

(2) Once a complete set of records is available :

iterative processing of parent fields , format and write output

Shared memory parallelism

Algorithm parallelization (inner loop parallelism, marked with ——— below)



Within the current processing iteration for the current output :

For each operator in turn :

parallel computation when the same transformation is applied on multiple fields

or

parallel computation on multiple partitions of the horizontal domain

Fieldextra – Performances (1)

- **Configuration for following performance results**
  - **Fieldextra 10.5.3**
  - Code compiled with **gfortran** with **-O3** optimization level
  - One Cray XE6 node (4x 2.1 GHz **AMD MagnyCour processors**, for a total of 24 cores)
  - **Lustre** parallel filesystem

  - Report total elapsed time (*tot*), time for decoding input (*in*), time for product generation (*prod*) and memory high water mark (*hwm*)

Fieldextra – Performances (2)

- **24h COSMO-1**, hourly GRIB 2 output with DD & FF on all levels
  - COSMO-1 , grid size **1062 x 774 x 80**
  - Input size about **60GB**, output size **6.6GB**, **25** products
  - 24h COSMO-1 production time is about **42'**

1x1 thread	**14'** (*tot*) **= 4'** (*in*) **+**	**10'** (*prod*)	**1.6 GB** (*hwm*)
1x6 threads	**10'** (*tot*) **= 4'** (*in*) **+**	**6'** (*prod*)	**1.7 GB** (*hwm*)

- **72h COSMO-7,** 1200 products (operational)
  - COSMO-7 , grid size **393 x 338x 60**
  - Input size about **23GB**, output size about **10GB**, about **1200** products
  - 72h COSMO-7 production time is about **30'**

1x1 thread	**25'** (*tot*) **= 5'** (*in*) **+**	**19'** (*prod*)	**+ 1'** (*other*)	
6x4 threads	**12'** (*tot*) **= 5'** (*in*) **+**	**6'** (*prod*)	**+ 1'** (*other*)	**4.4GB** (*hwm*)

Fieldextra – Performances (3)

- **Speedup between release 10.5.1 and 10.5.3 (optimization + OpenMP)**
  - **COSMO-7** production:
    from >3200 [s] to 720 [s] Speedup about **4.5** (6x4 threads)
  - **FABEC** production:
    from 4400 [s] to 280 [s] Speedup about **15** (1x6 threads)
  - **CAPE_MU** production on COSMO-7 domain:
    from 93 [s] to 22 [s] Speedup about **4.5** (1x6 threads)

- **Poor performance of lateral regridding with gfortran compiled code fixed in 10.5.2**
  - Problem was much less acute with Pathscale compiled code

- A **weak scalability** of fieldextra is obtained when the GRIB decoding time is neglected
  - Typically the situation of a production environment, where the size of the model output remains constant, but the number of products increases with the time

What shall I expect next?

Next public release

- **Releases 11.0 (→ Nov. 2012)**
  - New **operators**
    - geostr. vorticity, vorticity advection, thickness advection
    - wind divergence, humidity convergence
    - frontogenese function, CAT index
    - Interpolation on theta surfaces
  - Consolidated **test environment**
    - With support of H.Asensio / DWD
  - **Cookbook** with real life examples

- The release 11 will fulfill all the requirements defined at the postprocessing workshop (Langen, 26.02.2009)

What shall I expect next?
GRIB2 coordination

- **Short names**
  - Master table on COSMO web site, provided by DWD (Excel table)
  - Tool to derive fieldextra dictionary from master table

- **Model name** (no WMO standard mechanism)
  - Derived from the following set of keys
    center / subCenter / productDefinitionTemplateNumber / generatingProcessIdentifier
  - Each COSMO member define a unique combination of these keys for each model operated at their center and use them consistently, this is documented on the web
  - *fieldextra already supports this mechanism*

- **Experiment tag** (no WMO standard mechanism)
  - 'localNumberOfExperiment‘ as compulsory entry in all local use sections
  - Default local use sections (local.<centre>.250)
  - *fieldextra already supports this mechanism*

- **Local usage** (local use section, local usage in tables, local tables)
  - Usage description on COSMO web site

Beyond release 11.0

- **Priorities and resources not yet defined !**

- **Add functionalities to allow usage of fieldextra for COSMO-DE-EPS**
- **Version light without license fees for SRNWP-I**
- **Consolidate ASCII output** (e.g. uniform improved header, code clean-up)
- **Support new COSMO developments** (e.g. tiles, snow model)

- Add or consolidate support for additional products (e.g. radar, pseudo-satellite)
- Set of small improvements for COSMO-LEPS
- Wrapper scripts to offer simplified usage for common tasks (e.g. cropping)
- Finalize developer documentation

- *Parallel input*
- *NetCDF input*
- *Support ICON grid*

Thank you for your attention!

```
!+****************************************************************************
SUBROUTINE generate_output(multi_pass_mode, just_on_time, last_call,          &
                  datacache, data_origin, tot_nbr_input,                  &
                  out_paths, out_types, out_modes,                        &
                  out_grib_keys, out_spatial_filters,                     &
                  out_subset_size, out_subdomain, out_gplist, out_loclist, &
                  out_data_reduction, out_postproc_modules,               &
                  nbr_gfield_spec, gen_spec, ierr, errmsg                 )
!==============================================================================
!
! Root procedure to generate output files
!
!------------------------------------------------------------------------------
 ! Dummy arguments
 LOGICAL, INTENT(IN)                        :: multi_pass_mode      ! Multiple pass mode?
 LOGICAL, DIMENSION(:), INTENT(IN)          :: just_on_time         ! True if prod. now
 LOGICAL, INTENT(IN)                        :: last_call            ! True if last call
 CHARACTER(LEN=*), INTENT(IN)               :: datacache            ! Data cache file
 TYPE(ty_fld_orig), INTENT(IN)              :: data_origin          ! Data origin
 INTEGER, DIMENSION(:), INTENT(IN)          :: tot_nbr_input        ! Expected nbr. input
 CHARACTER(LEN=*), DIMENSION(:), INTENT(IN) :: out_paths            ! Output files names
 TYPE(ty_out_spec), DIMENSION(:), INTENT(IN) :: out_types           !   types
 TYPE(ty_out_mode), DIMENSION(:), INTENT(IN) :: out_modes           !   modes
 INTEGER, DIMENSION(:,:), INTENT(IN)        :: out_grib_keys        !   grib specs
 INTEGER, DIMENSION(:), INTENT(IN)          :: out_subset_size      !   subset size
 INTEGER, DIMENSION(:,:), INTENT(IN)        :: out_subdomain        !   subdomain definition
 INTEGER, DIMENSION(:,:,:), INTENT(IN)      :: out_gplist           !   gp definition
 CHARACTER(LEN=*), DIMENSION(:,:), INTENT(IN) :: out_loclist        !   locations definition
 CHARACTER(LEN=*), DIMENSION(:,:), INTENT(IN) :: out_spatial_filters ! Condition defining filter
 TYPE(ty_out_cmd), DIMENSION(:), INTENT(IN) :: out_data_reduction   ! Data reduction spec
 CHARACTER(LEN=*), DIMENSION(:), INTENT(IN) :: out_postproc_modules ! Specification of processing
 INTEGER, DIMENSION(:,:), INTENT(IN)        :: nbr_gfield_spec      !+ Specifications of
 TYPE(ty_fld_spec_root), DIMENSION(:), INTENT(IN) :: gen_spec       !+ fields to generate
 INTEGER, INTENT(OUT)                       :: ierr                 ! Error status
 CHARACTER(LEN=*), INTENT(OUT)              :: errmsg               !   error message

 ! Local parameters
 CHARACTER(LEN=*), PARAMETER           :: nm='generate_output: ' ! Tag

 ! Local variables
 LOGICAL                        :: exception_detected, exception, use_postfix
 LOGICAL                        :: unique_ftype, multiple_grid, exist
 LOGICAL, DIMENSION(3*mx_iteration+1)  :: tmp_fddata_alloc, tmp_gpdata_alloc
 LOGICAL, DIMENSION(3*mx_iteration+1)  :: tmp_value_alloc, tmp_flag_alloc
 INTEGER                        :: i1, i2, i3, i_fd, i_vd
 INTEGER                        :: nbr_input
 INTEGER                        :: out_idx, ios, idx_vd_defined
 CHARACTER(LEN=strlen)                 :: messg, temporal_res, out_path
 TYPE(ty_fld_type)              :: out_ftype



 ! Initialize variables
 !--------------------
 ierr = 0  ;  errmsg = ''
 exception_detected = .FALSE.
 tmp_fddata_alloc(:) = .FALSE.  ;  tmp_gpdata_alloc(:) = .FALSE.
 tmp_value_alloc(:) = .FALSE. ; tmp_flag_alloc(:) = .FALSE.


! Create/update data cache file
 !------------------------------------------------------------------
 ! The cache file must reflect the state of data(:) after the last call to
 ! collect_output (i.e. before any field manipulation done in prepare_pout)
```

```
! Loop over each output file
!--------------------------
output_file_loop: &
DO i1 = 1, nbr_ofile
  out_idx = data(i1)%ofile_idx
  nbr_input = COUNT( data(i1)%ifile_used )

  ! Skip bogus output
  IF ( data(i1)%ofile_bogus ) CYCLE output_file_loop
  ! Skip completed output
  IF ( data(i1)%ofile_complete ) CYCLE output_file_loop
  ! Skip empty data array
  IF ( ALL(.NOT. data(i1)%defined) ) CYCLE output_file_loop
  ! Only prepare output when all possible associated data have been collected
  ! or when 'just on time' production is active
  IF ( .NOT. last_call                 .AND.                 &
       nbr_input < tot_nbr_input(out_idx) .AND.              &
       .NOT. just_on_time(out_idx)          ) CYCLE output_file_loop

  ! At this point the corresponding output file will be produced
  ! Keep track of completed output file
  IF ( nbr_input >= tot_nbr_input(out_idx) ) data(i1)%ofile_complete = .TRUE.

  ! Build name of output, considering a possible temporary postfix
  use_postfix = .FALSE.
  IF ( LEN_TRIM(out_postfix) /= 0 .AND. data(i1)%ofile_usepostfix .AND.   &
       .NOT. (data(i1)%ofile_firstwrite .AND. data(i1)%ofile_complete)  )  &
                                     use_postfix = .TRUE.
  out_path = out_paths(out_idx)
  IF ( use_postfix ) out_path = TRIM(out_path) // out_postfix

  ! Release memory allocated in previous call to prepare_pout (if any)
  DO i2 = 1, 3*mx_iteration+1
    IF ( tmp_value_alloc(i2) ) DEALLOCATE(data_tmp(i2)%values, data_tmp(i2)%defined)
    IF ( tmp_flag_alloc(i2) ) DEALLOCATE(data_tmp(i2)%flag)
    IF ( tmp_fddata_alloc(i2) ) THEN
      DEALLOCATE(data_tmp(i2)%field_type, data_tmp(i2)%field_origin,        &
data_tmp(i2)%field_name, data_tmp(i2)%field_grbkey,        &
              data_tmp(i2)%field_trange,                    &
              data_tmp(i2)%field_level, data_tmp(i2)%field_ltype,        &
              data_tmp(i2)%field_prob, data_tmp(i2)%field_epsid,         &
              data_tmp(i2)%field_vref, data_tmp(i2)%field_ngrid,         &
              data_tmp(i2)%field_scale, data_tmp(i2)%field_offset,       &
              data_tmp(i2)%field_vop, data_tmp(i2)%field_vop_usetag,     &
              data_tmp(i2)%field_vop_nlev, data_tmp(i2)%field_vop_lev,   &
              data_tmp(i2)%field_pop, data_tmp(i2)%field_hop,            &
              data_tmp(i2)%field_top, data_tmp(i2)%nbr_level,            &
              data_tmp(i2)%level_idx, data_tmp(i2)%nbr_eps_member,       &
              data_tmp(i2)%eps_member_idx, data_tmp(i2)%field_idx        )
    ENDIF
    IF ( tmp_gpdata_alloc(i2) ) THEN
      DEALLOCATE(data_tmp(i2)%gp_coord, data_tmp(i2)%gp_idx,             &
              data_tmp(i2)%gp_lat, data_tmp(i2)%gp_lon, data_tmp(i2)%gp_h)
    ENDIF
  END DO

  ! Prepare data for print out (calculate new fields, ... ; populate data_pout)
  ! * Info message
  IF ( just_on_time(out_idx) ) THEN
    messg = ' (just on time output)'
  ELSE IF ( nbr_input >= tot_nbr_input(out_idx) ) THEN
    messg = ' (all associated input collected)'
  ELSE
    messg = ''
  ENDIF
```