WRF-Var Software
(*Version* 2.2-beta)

John Bray

January 2007

Outline:

- **Introduction**
- **Software Overview**
- **Data Structures**
- **Registry**
- **Example**

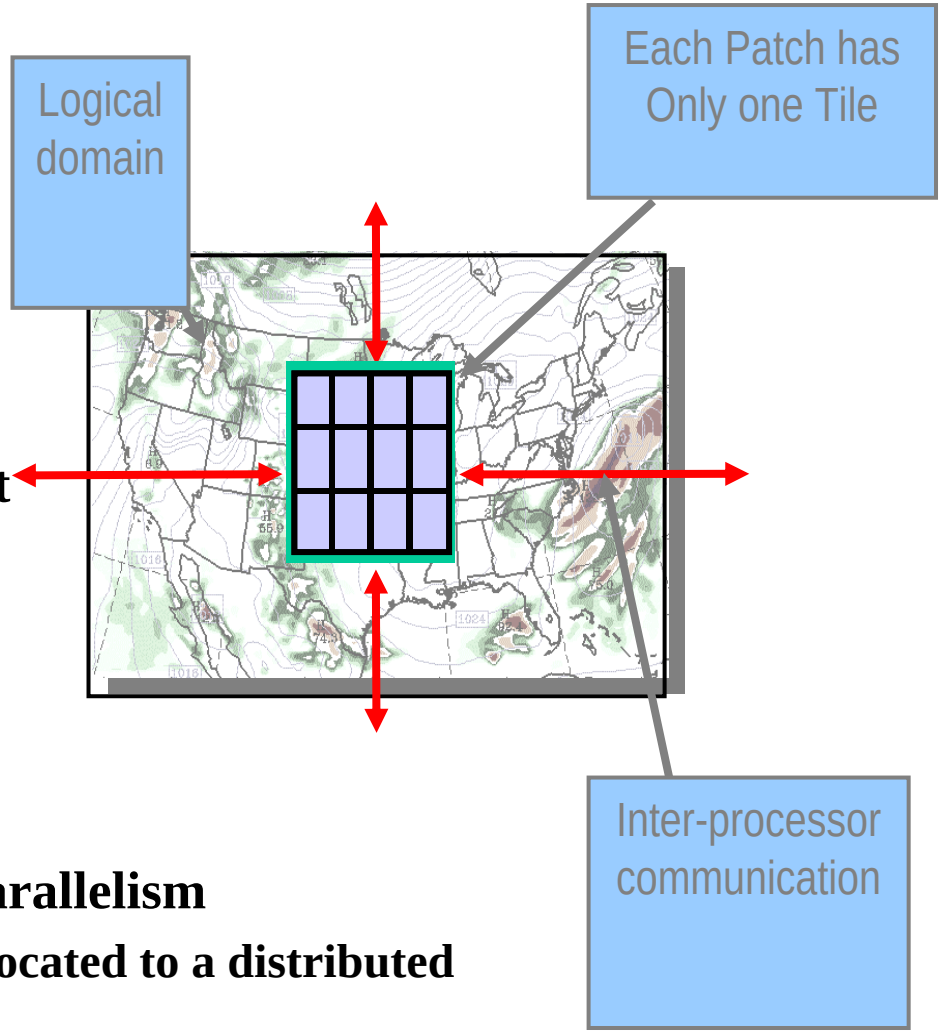
Introduction:

- **Intended audience for this tutorial session:**
  - **Primarily scientific users and others who wish to:**
    - **Work with the code**
    - **Extend/modify the code to enable their work/research**
    - **Address problems as they arise**
    - **Adapt the code to take advantage of local computing resources**
  - **Also: developers, computer scientists and software engineers, computer vendors**
    - **Developing new functionality (e.g. new observations, new minimization package)**
    - **Porting and benchmarking new platforms**

Supported Platforms:

- **IBM (AIX)**
- **HP (OSF1)**
- **MAC (OS X)**
- **PC (Linux)**
- **SGI (IRIX)**
- **CRAY (X1)**

Parallelism in WRF-Var: MPI Decomposition



- **Single version of code for efficient execution on:**
  - **Distributed-memory**
  - **Vector and microprocessors**

Model domain is decomposed for parallelism

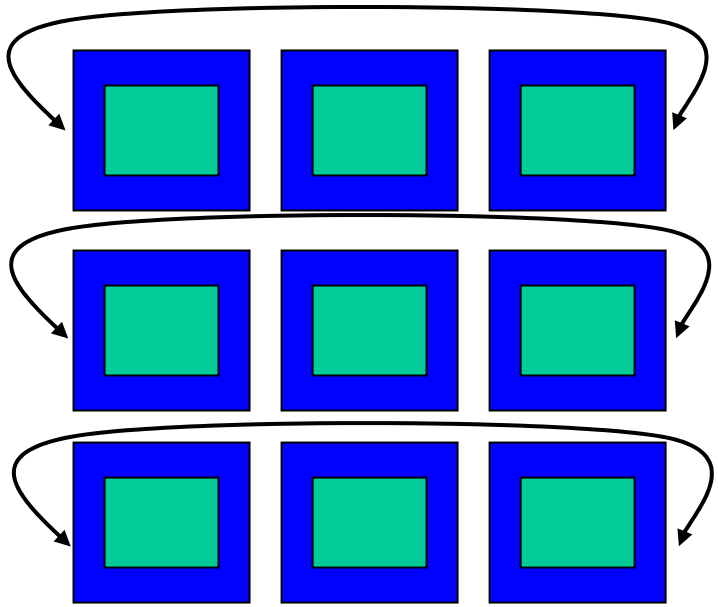
Patch: **section of model domain allocated to a distributed memory node**

Tile: **same as patch in WRF-Var**

Distributed Memory Communications

- **Halo updates**
- **Periodic boundary updates**

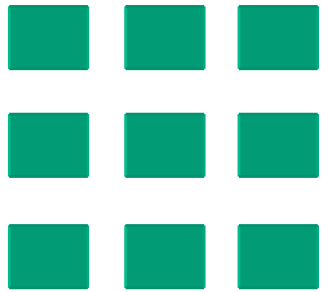
  (only needed for global 3dvar)



Distributed Memory Communications

- **Halo updates**
- **Periodic boundary updates**
- **Parallel transposes**
- **“nproc_x = 1”**

  (For global option)

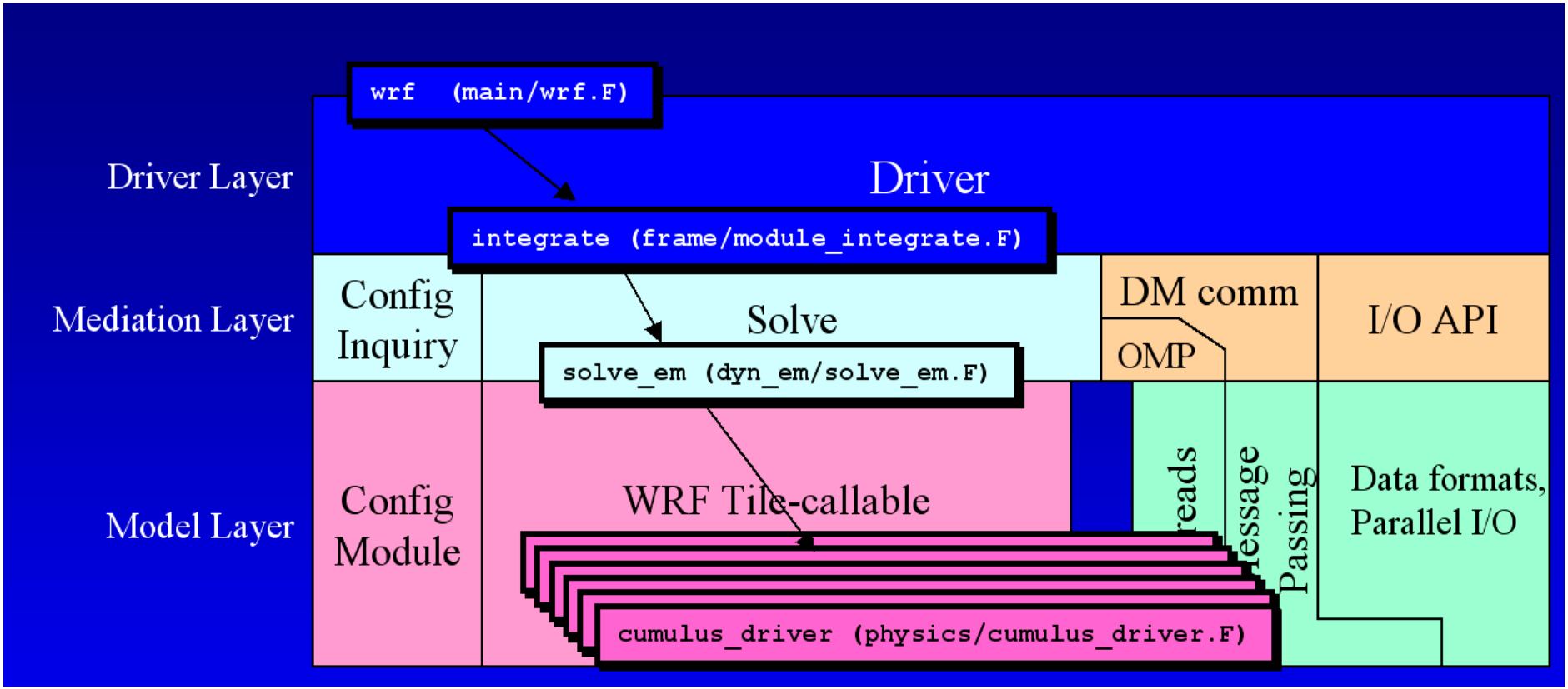


all y on patch

all z on patch

all x on patch

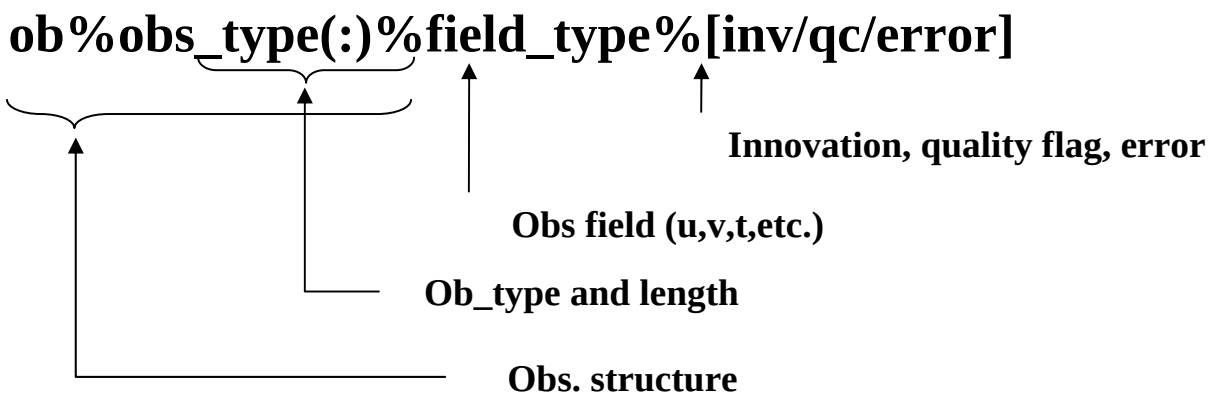
Directory Structure



WRF → WRF-Var
Integrate → wrf_3dvar_interface
Solve_em → da_solve_v3d
Cumulus_driver → obs. (DA_Ships) or DA_Minimisation etc.

WRF-VAR Observations

- **May be single level or multiple levels**
- **Ob_type or y_type:**



Example

Radiosonde observation appears as:

ob% sound(n)% u(lvl)% inc
ob% sound(n)% v(lvl)% qc
ob% sound(n)% v(lvl)% error

Radiosonde residual appears as:

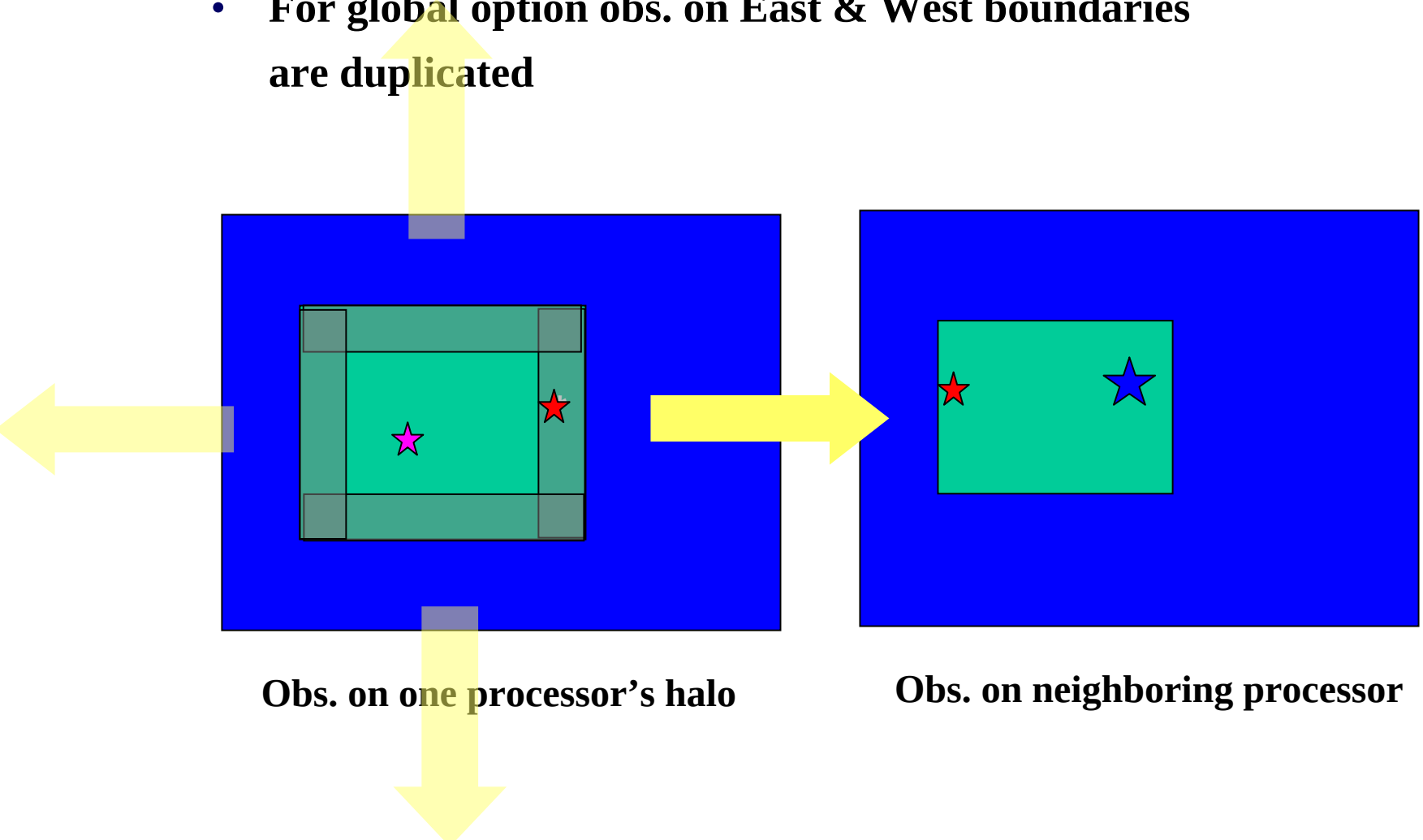
re% sound(n)% u(lvl)
re% sound(n)% v(lvl)

Observation Storage

- **Observations are stored in heap**
  - **Completely self-contained and private**
  - **Set once (Read in from disk file)**
  - **No exchange between processors/processes**

Observation in Distributed Memory

- **Halo Region Observation**
- **For global option obs. on East & West boundaries are duplicated**



Obs. on one processor's halo

Obs. on neighboring processor

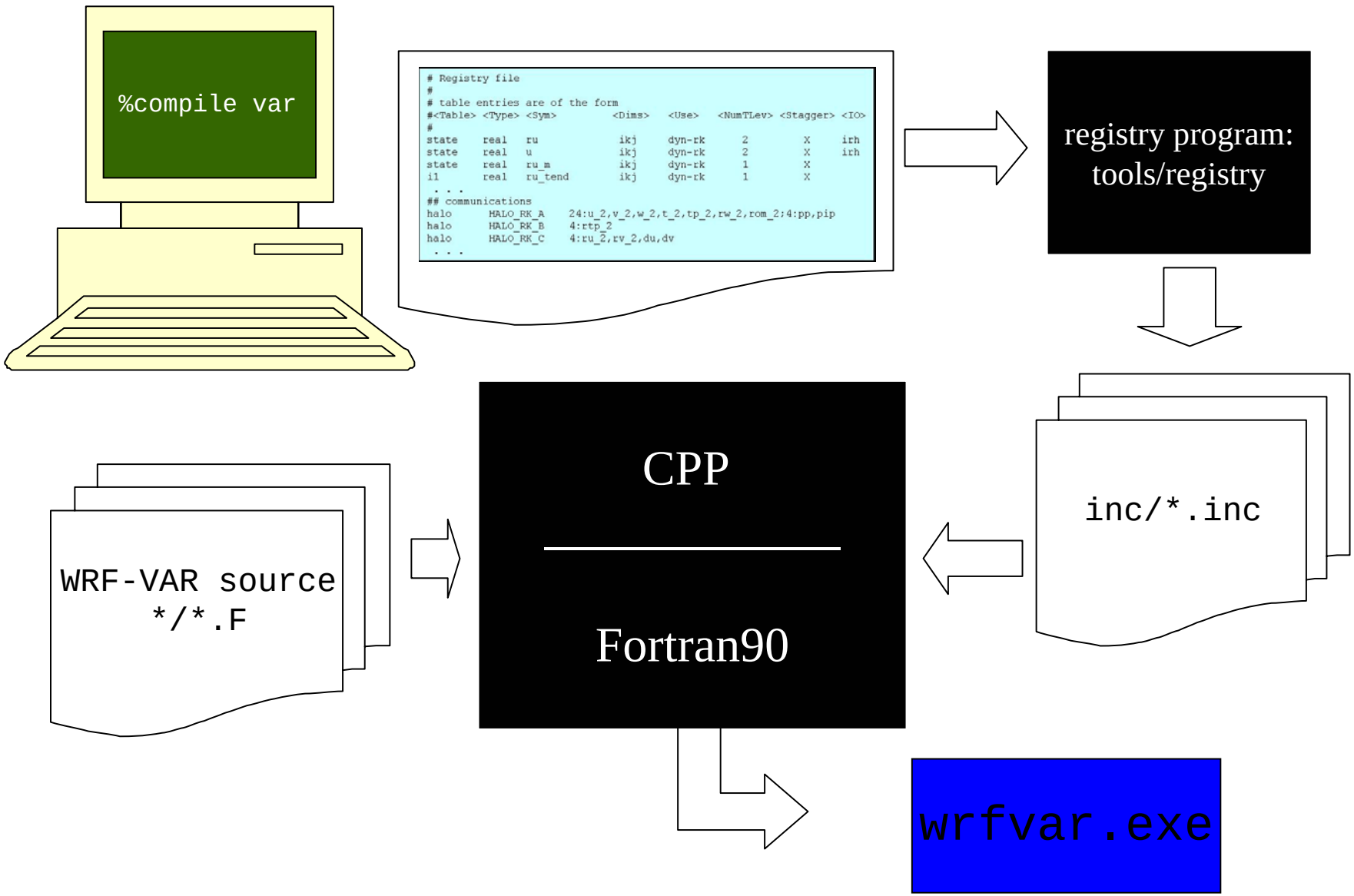
Grid Representation in Arrays

- **Increasing indices in WRF-Var arrays run**
  - **West to East (X, or I-dimension)**
  - **South to North (Y, or J-dimension)**
  - **Bottom to Top (Z, or K-dimension)**
- **Storage order in WRF-Var is IJK, but this is a WRF-Var convention, not a restriction of the WRF Software Framework**
- **WRF-Var grid data are all converted to mass-grid point**

WRF-Var Registry

- "Active data-dictionary" for managing WRF-Var data structures
  - Database describing attributes of model state, intermediate, and configuration data
    - Dimensionality, number of time levels, staggering
    - Association with physics
    - I/O classification (history, initial, restart, boundary)
    - Communication points and patterns
    - Configuration lists (e.g. namelists)
  - Program for auto-generating sections of WRF from database:
    - Argument lists for driver layer/mediation layer interfaces
    - Interprocessor communications: Halo and periodic boundary updates, transposes
    - Code for defining and managing run-time configuration information
- Automates time consuming, repetitive, error-prone programming
- Insulates programmers and code from package dependencies
- Allow rapid development
- Documents the data

Registry Mechanics



Registry Data Base

- **Currently implemented as a text file: Registry/Registry.3dvar**
- **Types of entry:**
  - ***State* – Describes state variables and arrays in the domain structure**
  - ***Dimspec* – Describes dimensions that are used to define arrays in the model**
  - ***Typedef* – Describes derived types that are subtypes of the domain structure**
  - ***Rconfig* – Describes a configuration (e.g. namelist) variable or array**
  - ***Halo* – Describes halo update interprocessor communications**
  - ***Xpose* – Describes communications for parallel matrix transposes**

State entry:

- **Elements**
  - ***Entry*: The keyword "state"**
  - ***Type*: The type of the state variable or array (real, double, integer, logical, character, or derived)**
  - ***Sym*: The symbolic name of the variable or array**
  - ***Dims*: A string denoting the dimensionality of the array or a hyphen (-)**
  - ***Use*: A string denoting association with a solver or 4D scalar array, or a hyphen**
  - ***NumTLev*: An integer indicating the number of time levels (for arrays) or hyphen (for variables)**
  - ***Stagger*: String indicating staggered dimensions of variable (X, Y, Z, or hyphen)**
  - ***IO*: String indicating whether and how the variable is subject to I/O and Nesting**
  - ***DName*: Metadata name for the variable**
  - ***Units*: Metadata units of the variable**
  - ***Descrip*: Metadata description of the variable**
- **Example**

```
#      Type Sym  Dims   Use     Tlev Stag IO    Dname
Descrip
# definition of a 3D, two-time level, staggered state array

state  real u   ijk    dyn_em   2   X    irh   "U"  "X WIND
COMPONENT"
…
typedef xb_type real   u   ijk        -           1    -      -
…
state xb_type xb - -
```

Comm entries: halo

- **Elements**
  - ***Entry*: keywords "halo"**
  - ***Commname*: name of comm operation**
  - ***Description*: defines the halo operation**
    - **For halo: *npts:f1,f2,...[;npts:f1,f2,...]****
- **Example**

```
halo    HALO_XA    dyn_em 24:xa%u,xa%v,xa%q,xa%p,xa%t,xa%rho,xa%rh,xa%psfc,xa%qcw,xa%qrn,xa%qt
halo    HALO_XB    dyn_em 24:xb%u,xb%v,xb%w,xb%wh,xb%q,xb%p,xb%t,xb%rho,xb%rh,xb%psfc,xb%slp
```

WRF-Var I/O

- **Uses same WRF I/O API features**

Procedure for adding new Observations

- **Edit DA_Define_Structure.F to add new data type**
- **Make new observation sub-directory under “src”**
- **Develop desired programs like getting innovation vector, forward observation operator, tangent linear & its adjoint, gradient & cost function etc. in this new sub-directory.**
- **Input observation (update DA_Obs)**
- **Sometimes it might be needed to add certain grid arrays in Registry**
- **Link into minimization package (DA_Minimisation)**