MOLOCH4PALM Technical Architecture

Tony Christian Landi, Oxana Drofa, Giorgio Resci and Fabrizio Roccato

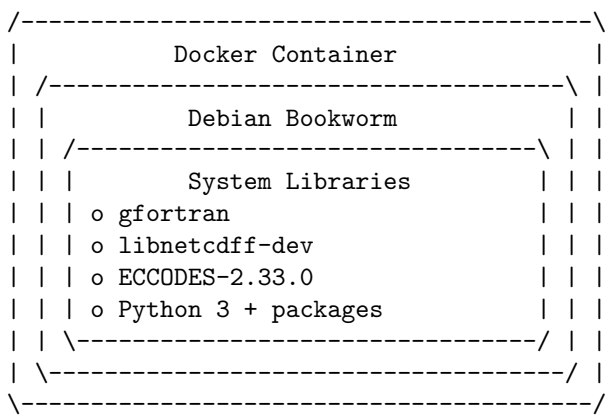
1 MOLOCH4PALM Technical Architecture

1.1 System Overview

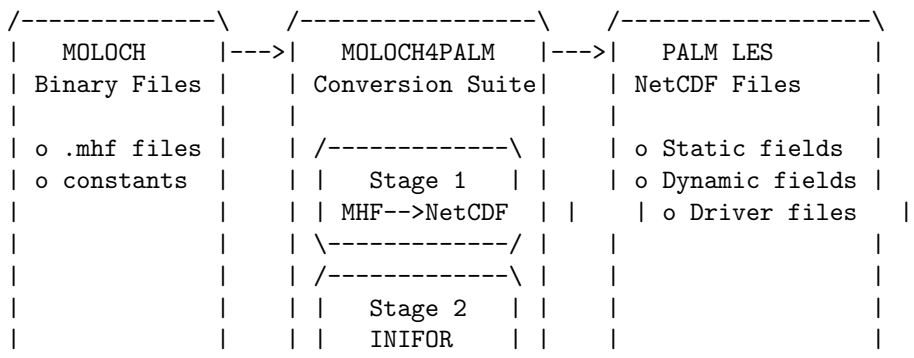
MOLOCH4PALM is a sophisticated data conversion pipeline designed to bridge two different atmospheric modeling systems. This document details the technical architecture and implementation.

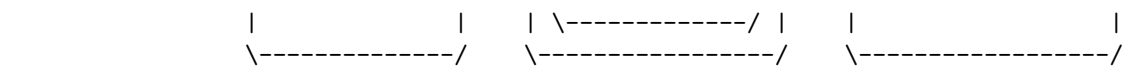
1.2 Architecture Components

1.2.1 1. Docker Container Infrastructure



1.2.2 2. Data Flow Architecture





1.3 Component Details

1.3.1 1. MOLOCH Data Processors

Legacy Processor (mhf2netcdf_2023_v0)

- **Language**: Fortran 90
- **Target**: MOLOCH data before June 30, 2024
- **Key Features**:
  Hourly output intervals only Basic variable set Original MHF format support

Current Processor (mhf2netcdf_2024_v1)

- **Language**: Fortran 90
- **Target**: MOLOCH data after July 1, 2024
- **Key Features**:
  Configurable minute-level intervals Extended variable set Enhanced metadata Improved CF compliance

1.3.2 2. Data Processing Modules

mod_moloch Module

```
module mod_moloch
  ! MOLOCH data structures
  integer :: nfdr(50)              ! Integer parameters
  real    :: pdr(100)              ! Real parameters

  ! Grid information
  real :: x0, y0                    ! Grid origin
  real :: dlon, dlat                ! Grid spacing
  real :: lon1, lat1                ! Corner coordinates

  ! Time information
  integer, dimension(5) :: idatec ! Current date
  integer :: idate0(5)             ! Reference date
end module
```

mod_palm Module

```
module mod_palm
  ! Domain parameters
  real :: lon_center, lat_center      ! Domain center
  real :: xlength_km, ylength_km      ! Domain size

  ! Output grid
  integer :: nlon_out, nlat_out, nlev_out
  integer :: jlon_ini, jlon_fin      ! Longitude indices
  integer :: jlat_ini, jlat_fin      ! Latitude indices

  ! Output arrays
  real, allocatable :: p_out(:,:,:)   ! Pressure
  real, allocatable :: t_out(:,:,:)   ! Temperature
  real, allocatable :: u_out(:,:,:)   ! U-wind
  real, allocatable :: v_out(:,:,:)   ! V-wind
  ! ... additional variables
end module
```

1.3.3 3. Processing Workflow

Stage 1: Binary to NetCDF Conversion

```
- Input: MOLOCH .mhf files
    |
  /-------------------------------------\
  | 1. Read MOLOCH header information   |
  |    o Grid dimensions (nlon, nlat)   |
  |    o Vertical levels (nlev, nlevg)  |
  |    o Time information               |
  \-------------------------------------/
    |
  /-------------------------------------\
  | 2. Read domain configuration        |
  |    o lon_center, lat_center         |
  |    o xlength_km, ylength_km         |
  |    o Calculate extraction indices   |
  \-------------------------------------/
    |
  /-------------------------------------\
  | 3. Process atmospheric data         |
  |    o Read 3D fields (p, t, u, v,    |
  |      w, q, qc)                      |
  |    o Calculate geostrophic winds    |
  |    o Apply domain extraction        |
  \-------------------------------------/
```

```
  |
/-------------------------------------\
| 4. Process surface/soil data        |
|    o Read 2D surface fields         |
|    o Read 3D soil fields            |
|    o Process radiation variables    |
\-------------------------------------/
  |
/-------------------------------------\
| 5. Write NetCDF output              |
|    o Create CF-compliant metadata   |
|    o Write coordinate variables     |
|    o Write data variables           |
\-------------------------------------/
  |
Output: NetCDF files ready for INIFOR
```

Stage 2: INIFOR Processing

```
Input: NetCDF files from Stage 1
  |
/-------------------------------------\
| INIFOR Preprocessor                 |
| o Reads PALM namelist               |
| o Interpolates to PALM grid         |
| o Generates boundary conditions     |
| o Creates initialization profiles   |
\-------------------------------------/
  |
Output: PALM dynamic driver file
```

1.4 Data Structures and Formats

1.4.1 MOLOCH Input Format (.mhf)

The MHF (Model History File) format is a Fortran binary structure:

```
Record 1: Header information
  - nfdr(50): Integer parameters array
  - pdr(100): Real parameters array

Record 2+: Time step data
  - Atmospheric variables (3D)
  - Surface variables (2D)
  - Soil variables (3D)
```

Key Variables Extracted: 3D Atmospheric Fields:

- Pressure (p): [Pa]
- Temperature (t): [K]
- U-wind component (u): [m/s]
- V-wind component (v): [m/s]
- Vertical velocity (w): [m/s]
- Specific humidity (q): [kg/kg]
- Cloud water content (qc): [kg/kg]

2D Surface Fields:

- Surface pressure (ps): [Pa]
- 2-meter temperature (t2): [K]
- Precipitation rates: [mm/h]
- Radiation fluxes: [W/m^2]

3D Soil Fields:

- Soil temperature: [K]
- Soil moisture: [m^3/m^3]

1.4.2 NetCDF Output Format

The system generates CF-compliant NetCDF files with:

Coordinate Variables:

```
lon(x): longitude [degrees_east]
lat(y): latitude [degrees_north]
lev(z): model levels [1]
time: time since reference [hours]
```

Data Variables:

```
P(time,lev,y,x): pressure [Pa]
T(time,lev,y,x): temperature [K]
U(time,lev,y,x): u_wind [m/s]
V(time,lev,y,x): v_wind [m/s]
W(time,lev,y,x): w_wind [m/s]
QV(time,lev,y,x): humidity [kg/kg]
QC(time,lev,y,x): cloud_water [kg/kg]
UG(time,lev,y,x): geostrophic_u [m/s]
VG(time,lev,y,x): geostrophic_v [m/s]
```

1.5 Coordinate Systems

1.5.1 MOLOCH Grid System

- **Projection**: Rotated latitude-longitude
- **Pole Position**: Variable (defined in model constants)
- **Grid Type**: Arakawa C-grid

1.5.2 PALM Grid System

- **Projection**: Regular latitude-longitude or local Cartesian
- **Grid Type**: Arakawa C-grid
- **Coordinate Origin**: User-defined

1.5.3 Transformation Process

```
! Example coordinate transformation
do jlat = 1, nlat
  do jlon = 1, nlon
    ! Convert from MOLOCH rotated coordinates
    call rotate_coordinates(lon_moloch(jlon), lat_moloch(jlat), &
                            lon_regular, lat_regular)

    ! Check if point is within PALM domain
    if (point_in_domain(lon_regular, lat_regular)) then
      ! Include in output
      include_point = .true.
    endif
  enddo
enddo
```

1.6 Memory Management

1.6.1 Allocation Strategy

```
! Dynamic allocation based on domain size
allocate(p_out(nlon_out, nlat_out, nlev_out))
allocate(t_out(nlon_out, nlat_out, nlev_out))
allocate(u_out(nlon_out, nlat_out, nlev_out))
! ... additional arrays

! Deallocate when done
deallocate(p_out, t_out, u_out)
```

1.6.2 Memory Optimization

- **Streaming I/O**: Processes one time step at a time
- **Domain Extraction**: Only allocates memory for output domain
- **Efficient Array Operations**: Uses Fortran array syntax

1.7 Error Handling and Quality Control

1.7.1 File Validation

```
! Check file existence
inquire(file='moloch_atm.mhf', exist=file_exists)
if (.not. file_exists) then
  print *, 'Error: Input file not found'
  stop
endif

! Check file size
inquire(file='moloch_atm.mhf', size=file_size)
if (file_size < expected_size) then
  print *, 'Warning: File may be incomplete'
endif
```

1.7.2 Data Validation

```
! Physical range checks
where (temperature < 200.0 .or. temperature > 400.0)
  temperature = missing_value
end where

! Consistency checks
if (maxval(pressure) < minval(surface_pressure)) then
  print *, 'Error: Inconsistent pressure fields'
endif
```

1.8 Performance Characteristics

1.8.1 Computational Complexity

- **Time**: $O(\text{N\_time} \times \text{N\_levels} \times \text{N\_horizontal})$
- **Space**: $O(\text{N\_horizontal\_out} \times \text{N\_levels})$

1.8.2 Typical Performance Metrics

```
Domain Size: 600km x 600km
Grid Points: 1200 x 1200 x 50 levels
Time Steps: 24 hours
Processing Time: ~10-15 minutes
Memory Usage: ~2-4 GB
Output Size: ~500 MB per time step
```

1.8.3 Optimization Techniques

1. **Vectorized Operations**: Uses Fortran array operations
2. **Efficient I/O**: Minimizes file read/write operations
3. **Memory Layout**: Optimizes array access patterns
4. **Domain Decomposition**: Processes spatial subdomains

1.9 Build System

1.9.1 Compilation Process

```
# Makefile structure
FC = gfortran
FCFLAGS = -O3 -fdefault-real-8
LDFLAGS = -lnetcdff -lnetcdf

# Object dependencies
moloch_to_palm.o: moloch_to_palm.F90 common_routines.o
common_routines.o: common_routines.F90

# Link executable
moloch_to_palm: $(OBJECTS)
    $(FC) $(LDFLAGS) -o $@ $^
```

1.9.2 Library Dependencies

- **NetCDF-Fortran**: Data I/O operations
- **ECCODES**: GRIB format support (optional)
- **Standard Libraries**: Mathematical functions

This technical architecture provides the foundation for understanding and maintaining the MOLOCH4PALM system.