MM5 Model Code (Appendix B)

- Code features
- Portability
- Vectorization
- Parallelization
- Use of pointers
- Code flow outline

NCAR/MMM

Code features

- More than 220 subroutines
- Selective compilation for physics and nesting
- 50 directories sorted by function (see end of Chapter 8)
- More than 55000 lines
- Standard Fortran 77 plus “Cray” pointers
- CPP #ifdef and #include commands
- Shared-memory parallel directives
- Distributed-memory extension library

NCAR/MMM

Portability

Shared memory or single processor

- Cray
- SGI
- Sun
- Compaq/DEC
- HP
- IBM
- Linux

Distributed memory (next talk)

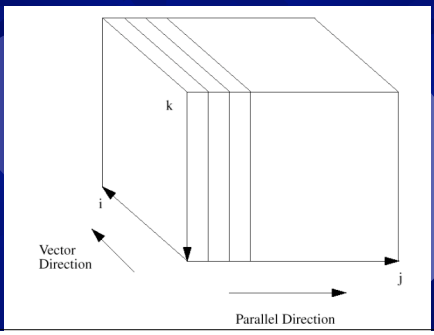
NCAR/MMM

Vectorization

- Originally written for Cray
- Vectorized over I index
- Inner loop (y direction)
- Solver-called routines operate on (I ,K) slices
- Only some physics options are vectorized

NCAR/MMM

Vector and parallel directions



Parallelization

- Shared-memory parallel directives
- Solver has outermost J loops
- Multi-tasking over J slices (each processor takes a J slice)
- Distinction made between *shared* and *private/local* variables
- No dependencies on results from other slices

NCAR/MMM

Parallelization (cont.)

- Local common blocks need special directives (*taskcommon* or *threadprivate)* to keep memory separate
- Multi-tasked loops have special directives (e.g. c$omp) ahead of them
- These are only seen as comments by non-parallel Fortran compilers

NCAR/MMM

Parallel loop directives

```
cmic$ do all autoscope
c$doacross
c$& local(i,j,k)
c$omp parallel do default(shared)
c$omp&private(i,j,k)
      DO J=1,JL
       DO I=1,ILX
        QDOT(I,J,1)=0.
        QDOT(I,J,KLP1)=0.
        W3DTEN(I,J,KLP1)=0.
      ENDDO
```

NCAR/MMM

Pointers

- A Cray feature that is now widely accepted by Fortran compilers
- MM5 uses pointers to handle nests
- Arrays in COMMON blocks appear as
  COMMON/ADDR1/IAUA, IAUB, IAVA, ...
  POINTER (IAUA, UA(MIX,MJX,MKX)), ...
- *IAUA* is an address for array *UA*
- MM5 has about 300 such addresses for all arrays/variables/constants associated with a single domain

NCAR/MMM

Pointers (cont.)

- 2 "super-arrays" *ALLARR* and *INTALL* store all these arrays end-to-end
- Additional super-arrays for FDDA
- Super-arrays are 2D, second dimension is domain id number (1 to *MAXNES*)
- 2D array *IAXALL* stores pointers (*IAUA, etc.*)
- Routine *ADDALL* sets pointer values in *IAXALL*
- Routines *ADDRX1C/ADDRX1N* reset pointers to different domains using *IAXALL.*
- Pointers are used in all the main COMMON blocks

NCAR/MMM

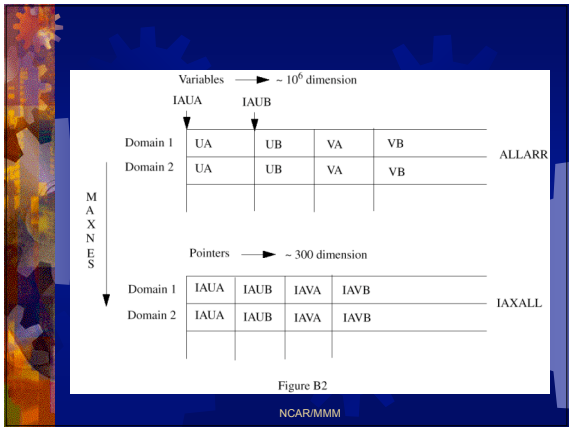


Figure B2

NCAR/MMM

Pointers (cont.)

- Pointers *were* also used to handle boundary I/O
- MM5 arrays are all dimensioned *(MIX,MJX,...)* using maximum dimensions of any nest
- However I/O needs actual domain dimension *(IX,JX,...)* to read in boundary file
- Pointers are used to mimic local variable-dimensioned arrays *(IX,JX,...)* that standard F77 does not allow
- **Note:** This has been discontinued with Version 3 format (1D scratch array read now)

NCAR/MMM

Code flow

- Main program is *Run/mm5.F*
  - Initialization
  - Main time loop for solver
  - Calls first-level nest driver
- Main solver is *dynamics/nonhydro/solve.F*
  - Calls dynamics and physics routines
  - Advances one domain by one timestep
- Nest driver is *domain/drivers/nstlev1.F*
  - Calculates nest boundary tendencies
  - Calls solver for nest 3 times
  - Calls next-level nest driver
  - Finally calls feedback for nest

NCAR/MMM