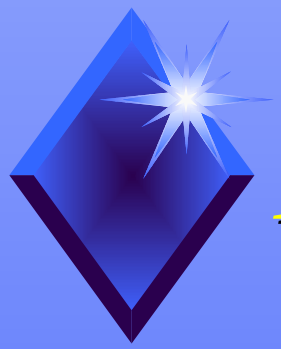
Strategies & Obstacles in Converting a Large Production Application to FORTRAN 90

David J. Gigrich
May 19, 1999
Structural Analysis Computing
The Boeing Company
david.j.gigrich@boeing.com

Topics of Discussion

- Intro. - Rationale for converting to FORTRAN 90
- Advantages
- Migration strategies
- Debugging techniques
- Obstacles / examples / clean-up
- Tools used
- System verification
- Resources & conclusions

BOEING



Rationale for Converting to f90

- Poor performance of PV+ CPUs
- CMOS technology - Gigaflop performance
- Phaseout of T90s and support for them
- Availability of spares - reliability
- Vendor limited support of FORTRAN 77

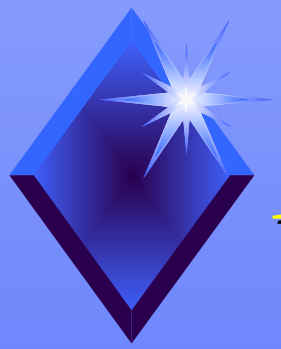
Advanatage of FORTRAN 90

- FORTRAN 90 is vendor standard
- Upward compatible libraries & object files
- Dynamic memory allocation
- Array operations (syntax)
- More intrinsic procedures
- Derived data types

BOEING

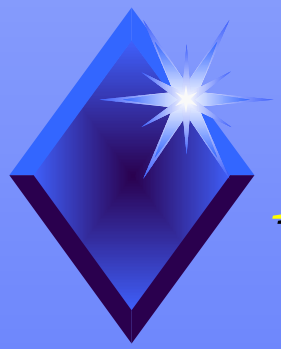
Advantages (continued)

- Performance improvements
  - Improved vectorization
  - Unrolling of loops
  - More in-lining of code
- Software can be simplified
- Reduced maintenance cost
- Improved portability



Migration Strategies

- Talk with other sites already using f90
- Convert small-modern applications first
- Verify external f90 libraries
- Use FORTRAN 90 compiler to locate noncompliances
  - Triton
  - Workstations (IBM RS6000, HPs)



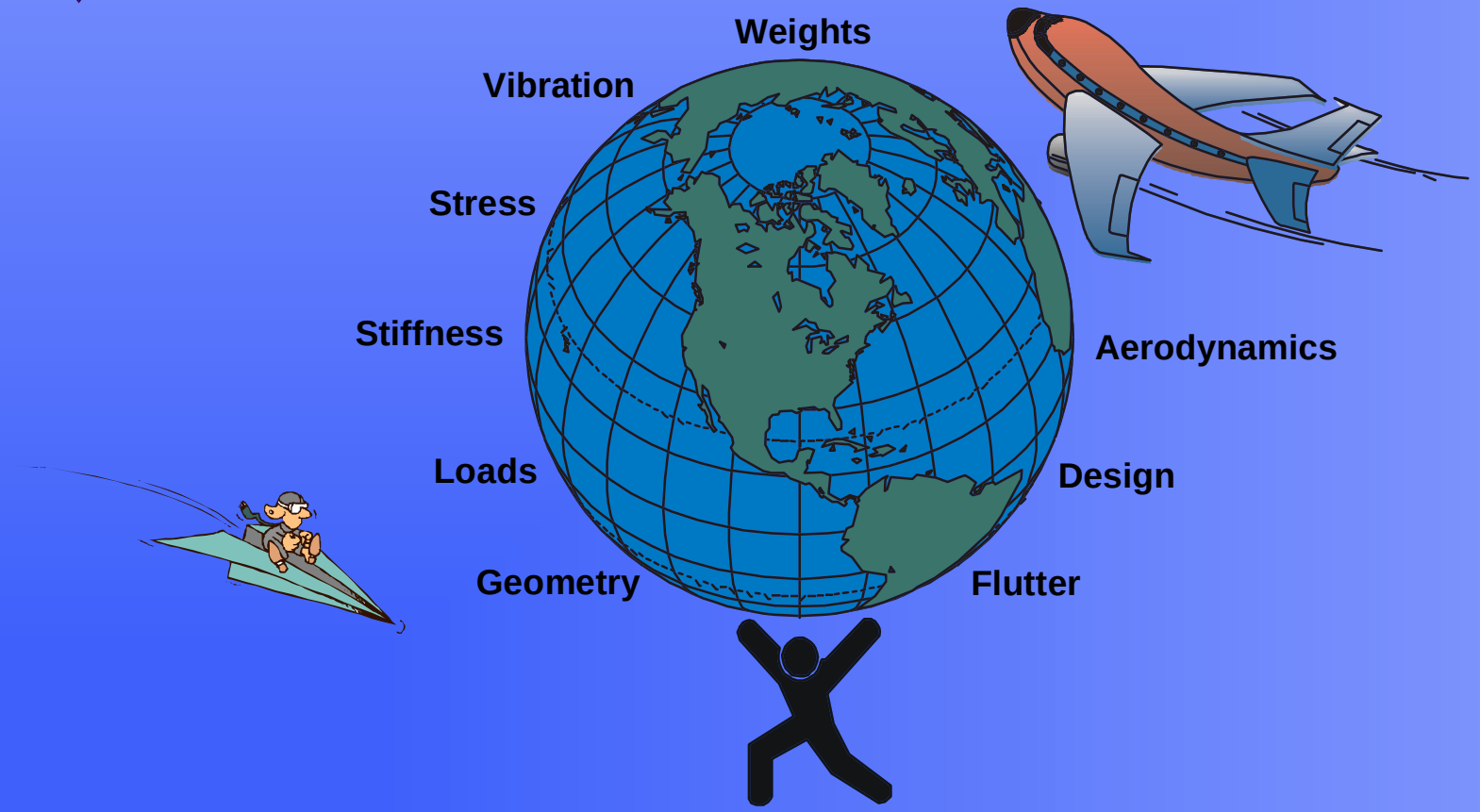
Migration Strategies (continued)

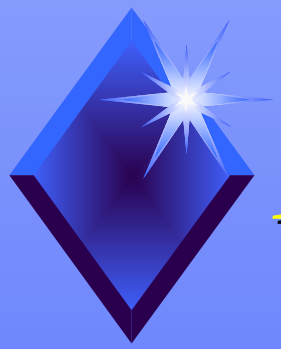
- Subdivide large applications
  - Support libraries
  - Selective loading/testing
  - Precompilers
  - Preprocessors, processors, postprocessors
  - Utilities (e.g. third party interfaces)
- Set number of CPUs 1

BOEING

ATLAS

An Integrated Structural Analysis and Design System
(1.4 Million Lines of Code)

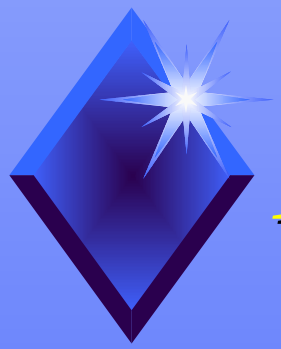




Debugging Techniques

- Address one problem at a time
- Interactive debugger (Totalview) for aborts
- Try to duplicate problem on cft77 system
  - Successful
    - Our code changed or
    - System libraries changed
- Isolate the f90 routine
  - Mix of f90 and cft77 objects

BOEING



Debugging Techniques (cont.)

- Restrict or eliminate optimization
- Check incoming & outgoing arguments
- Split routine in question into several
  - Mix of f90 and cft77 objects
- Use Totalview
  - Step through f90 version
  - Compare with f77 version

BOEING

Obstacles Encountered (Examples)

```
dimension a(10,3), b(9), c(10,9)
                .
call vecadd (a ,10 ,b ,3)
                .
                .
call vecadd (c, 10, b, 6)
-------------------------------------------------------
subroutine vecadd (x, nrow, b, num)
dimension x (nrow,3), b (num)
do i=1, num
   x (nrow, i) = x (nrow, i) + b (i)
enddo
return
```

Examples (optimization problems)

dimension a(1) <> a(*) <> a(n)

dimension b(n,1) <> b(n,*) <> b(n,m)

dimension c(n,3) <> c(n,*) <> c(n,m)

dimension d(1) <> d(n*m) <> d(n,m)

dimension e(n,m,1) <> e(n,m,*) <> e(n,m,k)

Examples (optimization continued)

```
dimension ifile(1)
equivalence( ifile, arnf)
common / kqrndm / arnf, brnf, ... , zrnf
```

```
do i = 1, n                <>   call dropfil (arnf, n)
   close (ifile (i) )           where:
enddo                           subroutine dropfil(ifiles,n)
                                dimension ifiles (n)
                                do i=1, n
                                   close ( ifiles (i) )
                                enddo
```

Examples (optimization continued)

```
dimension a (n,m), b (n*m)

do i= 1, n*m
   a ( i ,1) = b(i)                    | unpredictable results
enddo

do j =1 , m                                    do j=1,m
   do i = 1, n                          <>         a(:,j) = b((j-1)*n+1:)
      a ( i, j ) = b ( i + (j-1)*n )               enddo
   enddo
enddo
```

More Obstacles Encountered

- Loop error for variables with L - format (e.g. 3Labc ... 32 bit loop register)
- keybig = -mask(1) <> JMHCON(3)
- Missing routine arguments
- call writms (ntp8, nsizeb, 240, 3)
  - call writms (ntp8, nsizeb, 240, 3, irr)
  - call writms (ntp8, nsizeb, 240, 3, -1, 0)

Obstacles Encountered (compiler)

- Formats
  - 3x5e16.8 <> 3x, 5e16.8
- Dimension na (nxt, 5)
  - int = na (nxt) <> int = na (nxt, 1)
- Round-off differences
- Different variable memory locations
- Common block ordering

Examples (Mixed Arrays)

cft77

```
equivalence (cntrl(1), icntrl(1))
rval = icntrl (5) .or. 0
rval = rmove ( icntrl(5) )
rval = cntrl (5)
ival = rval
```

f90

```
rval = transfer ( icntrl(5), rval)
ival = rval
```

f90

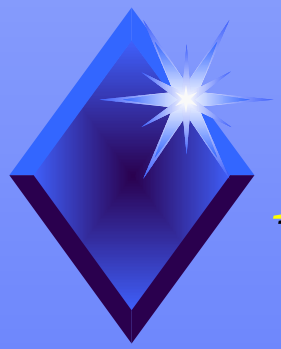
```
ival = transfer ( icntrl(5), rval)
```

Examples (clean - up)

- General format clean-up
  - O22 <> I9
  - nL <> nH
- Change pointers to allocatable arrays
- Replace loops or routine calls with f90 syntax where practical
- Automatic array allocations

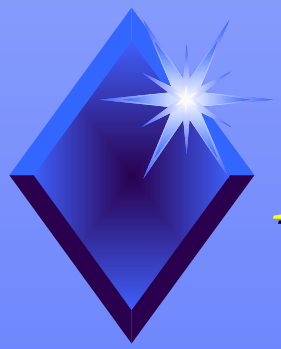
Tools Used / Developed

- Internal program to process SCCS files
  - (1), (x, 1)
  - dimension, real, integer, complex
- grep (0L, 1L, 2L, ... 9L)
- f90 Compilers (Triton and RS6000)
- SCCS
- Cflist & Totalview



Regression Testing

- Component testing
  - Libraries (system, data center, internal)
  - Preprocessors, postprocessing, processors
- 256 validation cases out of 443
- Continuious developer and user testing (over 11 months)
- Block point release validation (190 cases)



Resources

- Flow time: March 1998 to February 1999
- Labor - Hours:
  - 370 Analyst
  - 95 Engineering
- 2302 routines of 6423 modified
- Triton T916 with 512 MW
- Minimal machine resource impact

Conclusions

- Not tested . . . It won’t work !
- Sucessful conversion
  - Code executes more efficiently
  - Discovered many underlying array size errors
  - Applications are now more portable
  - Cost reductions
    - Maintance (do more with f90 and easier)
    - Development