THEORY IV

F90 COMPILER:

The most important switches

Important switches

F90 and C++

-C aopt (only C++) agressive vectorization and optimization (e.g. netcdf)
-C hopt Full use of vectorization and optimization
-C vopt Standard use of vectorization and optimization
-C sopt Full use of optimization in scalar code
-C vsafe Very safe use of vectorization and optimization
-C ssafe Safe use of optimization in scalar code
-C noopt (only C++) almost all optimizations and vectorization are surpressed
-C debug No vectorization and optimization at all

Important switches F90

-c	*compile only*
-ew	*Define basic storage unit as 8 bytes*
-eC	*Enable runtime array bounds checking*
-Ep	*Invoke Preprocessing*
-EP	*-Ep + save intermediate files*
-f0	*standard fixed format*
-f3	*extended fixed format*
-f4	*standard free format*
-ftrace	*flow trace profiling*
–g	*Generate program debugging tables*
-p	*Enable Profiling*

Word-Length

Please have always in mind

	Real	Real*4	Integer
Cray	8	8	8
SX-6/8	4	4	4
SX-6/8 (-ew)	8	8	8
SX-6/8(dbl4)	8	8	4
Workstation	4	4	4
WS (-r8)	8	4	4

Out of bounds references?

- f90 -eC
  - ‘array bounds checking’ enabled
- f90 -eC -Cvsafe
  - because otherwise:

```
real a(n,m)
...
Do j = 1, m
  do i = 1, n
    a(i,j) = 0.0
  end do
end do
```

Collapse →

```
real a(n,m)
...
...
do i = 1, n*m
    a(i,1) = 0.0
end do
...
```

Important switches F90

-P static	*Enable only vector processing (Default)*
-P auto	*Enable automatic parallel processing (not OpenMp)*
-P openmp	*Enable OpenMP parallel processing*
-P multi	*Enable parallel processing (local data allocated in task data memory)*
-P stack	*Enable parallel processing (local data allocated on stack)*
-pvctl vsqrt/novsqrt	*Hardware/Software Square Root on SX-8 (default: vsqrt)*

Important switches F90

pi		Specifies that the procedure inline function be used.
	auto	(default)Specifies automatic expansion.
	noauto	Specifies explicit inline expansion.
	line=*n*	Specifies the maximum number of lines (*n*) in a procedure for which automatic inline expansion is performed. The default value is 50.
	nest=*n*	Indicates that a procedure call is nested. The range of procedure subject to automatic inline expansion is limited by nesting level (*n)*. The default level is 1.
	exp=*routine-name* [,*routine-name*...]	Specifies that the specified routine name is expanded inline regardless of the specification of automatic expansion.
	noexp=*routine-name* [,*routine-name*...]	Specifies that the specified routine name is not expanded inline regardless of the specification of automatic expansion.
	rexp=*routine-name* [,*routine-name*...]	Specifies that the specified routine name, which is called nested expansion, is expanded in-line regardless of the specification of automatic expansion.
	expin= {*directory-name* | *file-name*} [,{*directory-name* | *file-name*}]	Specifies that the routine, which is expanded inline, is in the specified file or in Fortran source program files under the specified directory.

```
-A   Specifies to operate the automatic precision  expansion
         function. (The options -e w,  -float2  are ignored
         dbl    Specifies the precision expansion as R(4)->R(8),
                R(8)->R(16), C(4 )->C(8),and C(8) -> C(16).

         dbl4   Specifies the precision expansion as R(4)->R(8),
                and C(4)->C(8).

         dbl8   Specifies  the  precision  expansion  as   R(8)-
                >R(16), and C(8)->C(16).

         idbl   Specifies the precision  expansion  as  R(4)  ->
                R(8),  R(8)  ->  R(16), C(4) -> C(8) and C(8) ->
                C(16).
                However, a variable or a constant with kind type
                parameter  is  not  applied the precision expan-
                sion.

         idbl4  Specifies the precision  expansion  as  R(4)  ->
                R(8) and C(4) -> C(8).
                However, a variable or a constant with kind type
                parameter  is  not  applied the precision expan-
                sion.

          idbl8  Specifies the precision  expansion  as  R(8)  ->
                R(16) and C(8)
                 -> C(16).
                However, a variable or a constant with kind type
                parameter  is  not  applied the precision expan-
                sion.

    -NA  Specifies not to operate the automatic precision expan-
         sion function. (default)
```

Listing options

- -Wf" … "
  - Because it is passed to the compile phase
- -L fmtlist *formatted list*
- -L transform *transformations*
- -L map *reference listing*
- -L summary *detailed options used*
- Combine them!: -L fmtlist transform …

Optimization options F90

Sometimes –C hopt leads to problems. Then go back to –C vopt or
- try to identify the routine and
- switch off specific optimizations for this routine:

With –C hopt (default):

-i	inline intrinsic functions (default!)
-O nodiv	Division may not be changed (rec. multiplication)
-O nomove	Disable automatic movement of invariants outside of loop
-O nounroll	Disable automatic unrolling

...

...

...

OpenMP – Linkage with Fortran

```
f90 –Popenmp –c a.f

c++ -Popenmp –c b.cc

c++ -Popenmp –f90lib a.o b.o
```

```
f90 –Popenmp –c –ew a.f

c++ -Popenmp –c b.cc

c++ -Popenmp –f90libew a.o b.o
```

OpenMP – Runtime

Use Following Runtime Environment Variables

```
#!/bin/ksh

export OMP_NUM_THREADS=8
export F_PROGINF=DETAIL

openmp.x
```

ftrace and parallel execution

- single tasking:

```
PROG.UNIT  FREQUENCY  EXCLUSIVE       AVER.TIME   MOPS MFLOPS V.OP  AVER. I-CACHE O-CACHE    BANK
                      TIME[sec](  % )    [msec]               RATIO V.LEN  MISS    MISS      CONF

vdiff          30960    52.098( 13.5)     1.683 7997.0 3544.1 99.46 248.8  0.8383  0.2774  0.4611
progexp          720    46.437( 12.0)    64.496 7341.0 3389.2 97.90 121.0  2.1296  0.1366  0.0000
...
```

- multi 4 processors:

```
PROG.UNIT  FREQUENCY  EXCLUSIVE       AVER.TIME   MOPS MFLOPS V.OP  AVER. I-CACHE O-CACHE    BANK
                      TIME[sec](  % )    [msec]               RATIO V.LEN  MISS    MISS      CONF

vdiff          30960    54.823( 12.9)     1.771 7600.4 3367.9 99.46 248.8  0.9858  0.3081  0.4892
 -micro1        7952    14.175(  3.3)     1.783 7577.9 3358.2 99.46 249.7  0.2498  0.0765  0.1239
 -micro2        7806    13.841(  3.3)     1.773 7583.0 3360.1 99.46 248.5  0.2478  0.0775  0.1232
 -micro3        7786    13.771(  3.2)     1.769 7588.5 3362.3 99.46 248.1  0.2506  0.0788  0.1231
 -micro4        7416    13.036(  3.1)     1.758 7656.1 3392.6 99.46 248.8  0.2377  0.0753  0.1191
progexp         2880    50.762( 11.9)    17.626 6818.4 3146.6 97.79 121.1  2.6227  0.6504  0.0000
 -micro1         720    12.650(  3.0)    17.570 6877.1 3173.1 97.79 121.1  0.6162  0.0409  0.0000
 -micro2         720    12.753(  3.0)    17.713 6762.8 3121.0 97.79 121.1  0.6526  0.3125  0.0000
 -micro3         720    12.656(  3.0)    17.578 6816.6 3145.8 97.79 121.1  0.6735  0.1455  0.0000
 -micro4         720    12.702(  3.0)    17.642 6817.7 3146.6 97.79 121.1  0.6805  0.1514  0.0000
...
```

How to compile MPI-model

```
mpif90 mpicode.f90 -o mpicode

                    or (crosscompiler)

sxmpif90 mpicode.f90 -o mpicode
```

Don't link with -lmpi or include anything related to mpi, it is not necessary!

How to run a MPI-model

```
mpirun -np 2 a.out (arg1) (arg2)

mpirun -np 2 host1 -np 3 host2 a.out(arg1)(arg2)

                          or

mpiexec -n …
```

MPIPROGINF=DETAIL

How to run a MPI-model with configuration file

mpirun [*options*] { **-f** | **-machinefile** } <*configuration-file-name*>

How to run a MPI-model with configuration file

```
mpirun –f sx.conf
```

sx.conf:

```
-h sx6      -p 1 -e coupler.x
-h sx6      -p 1 -e appl4.x
-h sx6      -p 1 -e appl2.x
-h sx6      -p 1 -e appl1.x
-h sx6      -p 1 -e appl3.x
```

MPI Tracing

MPI Procedure Tracing Facility

Use -mpitrace
one can see the following information.

- which MPI procedure is called
- MPI process rank calling the MPI procedure
- entering and returning to/from MPI procedure

MPI: result of mpitrace

```
cs17 % doit 2
Starting MPI_Init...
Starting MPI_Init...
Starting MPI_Init...
Starting MPI_Init...
[0,0] Ending MPI_Init
[0,2] Ending MPI_Init
[0,1] Ending MPI_Init
[0,3] Ending MPI_Init
[0,1] Starting MPI_Comm_size...
[0,1] Ending MPI_Comm_size
[0,1] Starting MPI_Comm_rank...
[0,1] Ending MPI_Comm_rank
[0,0] Starting MPI_Comm_size...
[0,1] Starting MPI_Barrier...
[0,0] Ending MPI_Comm_size
[0,0] Starting MPI_Comm_rank...
[0,0] Ending MPI_Comm_rank
 PIP: number of processors:   4
[0,0] Starting MPI_Barrier...
[0,3] Starting MPI_Comm_size...
[0,3] Ending MPI_Comm_size
[0,3] Starting MPI_Comm_rank...
[0,3] Ending MPI_Comm_rank
[0,2] Starting MPI_Comm_size...
[0,3] Starting MPI_Barrier...
[0,2] Ending MPI_Comm_size
[0,2] Starting MPI_Comm_rank...
[0,2] Ending MPI_Comm_rank
[0,2] Starting MPI_Barrier...
[0,1] Ending MPI_Barrier
```

MPI Communication Information

Compile with

-mpiprof

Use Environment Variable
MPICOMMINF=(NO/YES/ALL)

```
MPI Communication Information:
------------------------------

Real MPI Idle Time (sec)    :        0.006 [0,0]        0.009 [0,1]        0.007
User MPI Idle Time (sec)    :        0.006 [0,0]        0.008 [0,1]        0.007
Total real MPI Time (sec)   :        0.047 [0,0]        0.068 [0,1]        0.058
Send       count            :            9 [0,0]            9 [0,0]            9
Recv       count            :            9 [0,0]            9 [0,0]            9
Barrier    count            :           14 [0,0]           14 [0,0]           14
Bcast      count            :            1 [0,0]            1 [0,0]            1
Reduce     count            :            0 [0,0]            0 [0,0]            0
Allreduce  count            :            0 [0,0]            0 [0,0]            0
Scan       count            :            0 [0,0]            0 [0,0]            0
Exscan     count            :            0 [0,0]            0 [0,0]            0
Redscat    count            :            0 [0,0]            0 [0,0]            0
Gather     count            :            0 [0,0]            0 [0,0]            0
Gatherv    count            :            0 [0,0]            0 [0,0]            0
Allgather  count            :            0 [0,0]            0 [0,0]            0
Allgatherv count            :            0 [0,0]            0 [0,0]            0
Scatter    count            :            0 [0,0]            0 [0,0]            0
Scatterv   count            :            0 [0,0]            0 [0,0]            0
Alltoall   count            :            0 [0,0]            0 [0,0]            0
Alltoallv  count            :            0 [0,0]            0 [0,0]            0
Alltoallw  count            :            0 [0,0]            0 [0,0]            0
Number of bytes sent        :         4088 [0,0]         4088 [0,0]         4088
Number of bytes recv        :         4088 [0,0]         4088 [0,0]         4088
Put        count            :            0 [0,0]            0 [0,0]            0
Get        count            :            0 [0,0]            0 [0,0]            0
Accumulate count            :            0 [0,0]            0 [0,0]            0
Number of bytes put         :            0 [0,0]            0 [0,0]            0
Number of bytes got         :            0 [0,0]            0 [0,0]            0
Number of bytes accum       :            0 [0,0]            0 [0,0]            0

[0,0] Ending MPI_Finalize
```

- S Generate Assembly listing

```
ident "FORTRAN90/SX Rev 290"
ident "'-S'"
ident "quiz1.f90"
 # Data section ordering define
        data
        align   16
 # ============ Begin  MAIN__ ============
 # === BSS SECTION ===
 #    == UNINIT LOCAL AREA ==
        bss     _.1.MAIN__.RTP.INTF_AREAPACKET, 0x0000001c0, 16
        bss     _.1.MAIN__.LOCALSTATIC.c, 0x000000190, 8
        bss     _.1.MAIN__.LOCALSTATIC.b, 0x000000190, 8
        bss     _.1.MAIN__.LOCALSTATIC.a, 0x000000190, 8
        set     .2.1.2spill_p, 0x000000130
 #  === TEXT SECTION ===
 #    == LINKAGE AREA ==
        text
        align   128
.3.H.LK.MAIN_:
 using  .3.H.LK.MAIN_,$s4
.2.1.2H.LK_p.1:
        llong   0xffffffffffffffff
        long    .3.H.CD.MAIN_-.3.H.LK.MAIN_
        long    25
        llong   0x0
        llong   .3.H.TB.MAIN_
        rlong   7,0
```

F90 Compiler Directives

Repeat the most important directives

F90 Compiler Directives Collapse

```
do i=1,n
 do j=1,m
   a(i,j)=b(i,j)+c(i,j)*x
 end do
end do
```

Tuned

```
do i=1,n*m
 a(i,1)=b(i,1)+c(i,1)*x
end do
```

F90 Compiler Directives

-Wf,-L fmtlist summary map transform

```
      !CDIR COLLAPSE
W ———— do i=1,n
 | |———— do j=1,m
 | |        a(i,j)=b(i,j)+c(i,j)*x
 | |———— end do
W ———— end do
```

Compiler directives F90: LOOPCHG

Example: n >> m

```
do i=1,n
  do j=1,m
    a(i,j) = b(i,j)+m(j)
  end do
end do
```

Compiler directives F90: LOOPCHG

```
      !CDIR LOOPCHG
X            do i=1,n
              do j=1,m
                a(i,j) = b(i,j)+m(j)
              end do
X            end do
```

But: Look for bank conflicts in PROGINF, if you do it manually

Compiler directives F90: UNROLL

```
do j=1,100
  … any stuff which prevents vectorization
  do i=1,n
    a(i) = a(i) + b(i,k) * c(k,j)
  end do
end do
```

Unrolling an outer loop that is not vectorized is generally effective, especially if it reduces the number of references to a memory area

Compiler directives F90: UNROLL

Example: Unroll by 2

```
!CDIR UNROLL=2
do j=1,99,2
  do i=1,n
    a(i) = a(i) + b(i,k) * c(k,j) + b(i,k)*c(k,j+1)
  end do
end do
```

The load count and store count of a(i) and the load count of b(i,k) is halved by unrolling

Compiler directives F90: NODEP

```
+------ DO J=1,N
|          X(J-1)=X(JW)*Y(J)
|
S          JW=JW+1
------ END DO
```

Investigate the values of JW
e.g with print the values in a list and look for duplicate indices

Optimization options F90

Please refer to

FORTRAN90/SX – Programmers Guide
Chapter 10 contains very useful informations

F90-Compiler Runtime Environment Variables

F_FILEINF=(YES/DETAIL/NO)
F_PROGINF=(YES/DETAIL/NO)
F_FTRACE=(YES/NO)
F_EXPRCW=unitnumber (>2 GB unform. output)*
F_FMTBUF=num (num bytes, 32 kbytes)
F_SETBUF=n kbytes (128 kbytes)

*Record Control Word is 8 byte storage
Additional Env. Var. in your handbook

COMPILER:

Investigation for correctness

Investigation of Correctness

- f90 -Cdebug
- f90 -eC (array bounds checking)
- f90 -eC -Cvsafe
- f90 -Pstack (default is -Pstatic)
- f90 -Pstack -Wf"-init stack=nan heap=nan"

- sxc++ -init stack=nan
- sxc++ -g

Code doesn't run (well?): f90 -Cdebug

- often already helps
  - switches off all optimisations
  - if it works: *not necessarily a compiler bug!*
- No core dump?
  - Nothing is worse than a code that runs but produces wrong results!
    - small test case!
    - 'trusted' reference output

Out of bounds references?

- f90 -eC
  - 'array bounds checking' enabled
- f90 -eC -**Cvsafe**
  - because otherwise:

```
real a(n,m)
...
Do j = 1, m
  do i = 1, n
    a(i,j) = 0.0
  end do
end do
```

→

```
real a(n,m)
...
...
do i = 1, n*m
    a(i,1) = 0.0
end do
...
```

Out of bounds references?

```
cs24 % a.out
    * 240 Subscript error  array=a size=100 subscript=101 eln=4
PROG=ec_test E
LN=4(400000908)
cs24 %
```

Variables initialised? Second example

- Typo on common block

```
common / .. / retv, valeu
real a(n,m)
do j = 1, m
  do i = 1, n
    retv = retv + a(i,j)
  end do
end do
retv = max( retv, value )
value = retv
```

- Use implicit none
- FORTRAN include files!

Variables initialised? The ultimate test

- f90 -Pstack -Wf"-init stack=nan heap=nan"
  - everything is initalised with the IEEE NaN
  - especially every local variable and array on call to a subroutine!
- Some overhead!
  - Do not use this for production
- for common blocks and other static data:
  - add to above: -Wl"-f nan"

C++ COMPILER:

The most important switches

C++ COMPILER:

Replaces also the C-Compiler !

C++ Compiler

C according to ANSI/ISO 9899:1990[1992], ISO/IEC 9899:1992 und ISO/IEC 9899:1990/Amd.1:1995 are supported,

C++ Compiler is compliant with ISO/IEC 14882-1998 and ISO/IEC 9899-1999

C++-Compiler

2.2.3 Inlining Options

-dir { inline | noinline }
-pi { auto | noauto }
{ inline | noinline }
max_depth=*n* (default: **max_depth=2**)
max_size=*n* (default: **max_size=50**)
Table 2-4 Inlining Options

Option	Suboption	Description	Default
-dir	**[no]inline**	Inlining directive options are [not] validated.	**noinline**
-pi		Enables inlining functions.	
	[no]auto	Enable[Disable] automatic inlining.	**noauto**
	[no]inline	**inline** specifiers in a C++ source program are [not] effective. This option is effective in the C++ mode.	**inline**
	max_depth =*n*	Specifies the level of functions to be inlined automatically from the bottom of the calling. This suboption is effective when **-pi,auto** is specified.	2
	max_size=*n*	Specifies the size of intermediate representation of functions to be inlined automatically. This suboption is effective when **-pi,auto** is specified.	50

C++-Compiler

C++-Compiler

2.2.6 Language Options
{ -Xp | -Xa | -Xc | -Xkr }
-K { const_string_literals | noconst_string_literals }
{ exceptions | noexceptions }
{ new_for_init | old_for_init }
{ restrict | norestrict }
{ rtti | nortti }
{ unsigned_char | signed_char }
{ using_std | nousing_std }
-T { auto | noauto }
{ none | all | used | local }
implicit_include
Table 2-7 Language Options

Option	Description	Default
-Xp	C++ mode. Enables compilation and linking of C++.	default
-Xa	ANSI C mode. Enables compilation and linking of C rather than C++.	-
-Xc	Strict ANSI C mode. Enables compilation and linking of C rather than C++.	-
-Xkr	K&R C mode. Enables compilation and linking of C rather than C++.	-

Important switches C++

-c *compile only*
-f90lib *Link with FORTRAN90/SX runtime libraries*
-math *Use mathematical intrinsic functions*
vector Use vector versions
scalar Use sclalar versions
-over2g Enable structures/arrays/classes/unions > 2GB
-restrict= Enable optimization and vectorization of loops with pointers and pointer references
all

C++-Compiler

Option	Suboption	Description	Default
-dir	**[no]inline**	Inlining directive options are [not] validated.	**noinline**
-pi		Enables inlining functions.	
	[no]auto	Enable[Disable] automatic inlining.	**noauto**
	[no]inline	**inline** specifiers in a C++ source program are [not] effective. This option is effective in the C++ mode.	**inline**
	max_depth=*n*	Specifies the level of functions to be inlined automatically from the bottom of the calling. This suboption is effective when **-pi,auto** is specified.	2
	max_size=*n*	Specifies the size of intermediate representation of functions to be inlined automatically. This suboption is effective when **-pi,auto** is specified.	50

C++-Compiler Vector Directives

almost similar to F90

#pragma cdir *vector_directive*

#pragma cdir nodep

C++ Compiler

Q: How are Fortran function names in the calling convention?

A: Simply write them lowercase and append an underscore.

Example:

```
PROGRAM TEST

CALL CFUNC(5)

END


void cfunc_(int *a) {

}
```

C++ Compiler

Q: How to link when C and Fortran are mixed?

A: Link with sxf90. It makes it right.

```
sxf90 fortran.o c++.o -o a.out
```

When using C++ as linker use:

```
sxc++ fortran.o c++.o -f90lib -o a.out
```

C++ Compiler, Porting Pitfalls

Q: I'm using malloc(), and my code dumps core.

A: Make sure stdlib.h is included where malloc() is used.

C++-Compiler

C++-Compiler

C++-Compiler

C++-Compiler

CPP-Options

f90 –Ep Invoke CPP tp perform preprocessing

f90 –EP Invoke CPP tp perform preprocessing and save intermediate file

CPP-Predefined Variables

-Uname

Unname the previously defined reserve symbol of the preprocessor. The current reserve symbols in SUPER-UX are "unix", "_FLOAT0", "_FLOAT1", "_FLOAT2", and "SX".

-Dname

-Dname=def

Define name with value def as if by a #define. If no =def is given, name is defined with the value 1. The -D option has lower precedence than the -U option. That is, if the same name is used in both a -U option and a -D option, the name will be undefined regardless of the order of the options. Characters from supplementary code sets can be used in def.

CPP-Predefined Variables, Example 1

```
program cpp_test
dimension SX(20),temp(20)
SX=10
do i=1,20
temp(i)=SX(i)
enddo
print*, temp(10)
end
```

```
cs24 % f90 -EP cpp_test.F
./i.cpp_test.F:

f90: error(106): i.cpp_test.F, line 3: Syntax error in assignment statement.
f90: error(110): i.cpp_test.F, line 5: Extra text follows the end of statement.
f90: warn(82): i.cpp_test.F, line 8: Name "sx" is not used.
f90: warn(81): i.cpp_test.F, line 8: Name "temp" is referenced but not defined.
f90: i.cpp_test.F, cpp_test: There are 2 errors and 2 warnings.
f90 fatal : /usr/lib/f90com command error : 1
cs24 %
```

CPP-Predefined Variables, Example 1

```
program cpp_test
dimension SX(20),temp(20)
SX=10
do i=1,20
temp(i)=SX(i)
enddo
print*, temp(10)
end
```

```
cs24 % f90 -EP -USX cpp_test.F
./i.cpp_test.F:

f90: vec(4): i.cpp_test.F, line 3: Vectorized array expression.
f90: vec(1): i.cpp_test.F, line 4: Vectorized loop.
f90: i.cpp_test.F, cpp_test: There are 2 diagnoses.
cs24 %
```

CPP-Predefined Variables, Example 2

```
 program cpp_test
     dimension sx(20),temp(20)
     sx=10
     do i=1,20
     temp(i)=sx(i)
     enddo
#if defined(SX)
     print*, 'This is an SX'
#else
     print*, 'This is no SX'
#endif
     end
```

sxf90 -EP cpp_test.F gives:

```
program cpp_test
    dimension sx(20),temp(20)
    sx=10
    do i=1,20
    temp(i)=sx(i)
    enddo

    print*, 'This is an SX'

    end
```

C++-Compiler Runtime Environment Variables

C_FILEINF= (YES/DETAIL/NO)

C_FTRACE= (NO/YES)

C_PROGINF=(NO/YES/DETAIL)

Debugging:

- xdbx, X-based dbx
- don't forget to set DISPLAY
- refer to manual
- compile and link with -C debug
- very similar to other vendors' debugger

Debugging:

For programs coded in C

% cc -g *myprog.c* **-o** *myprog*

For programs coded in FORTRAN

% f90 -Cdebug *myprog.f* **-o** *myprog*

```
% dbx myprog
dbx version X.XX of MM/DD/YY HH:MM.
Type 'help' for help.
reading symbolic information
(dbx)
```

Debugging:

```
(dbx) stop at 20
[1] stop at "myprog.":20
(dbx) stop at 25
[2] stop at "myprog.":25
(dbx)
```

Setting stops

```
(dbx) run
[1] stopped in MAIN_ at line 20 in file "myprog.f"*
20 INT4=3
21 (dbx)
```

```
(dbx) list 17, 20
   17    IF (INT1 .LT. 0) THEN
   18    INT4=0
   19    ELSE
   20    INT4=3
 (dbx) print int1
```

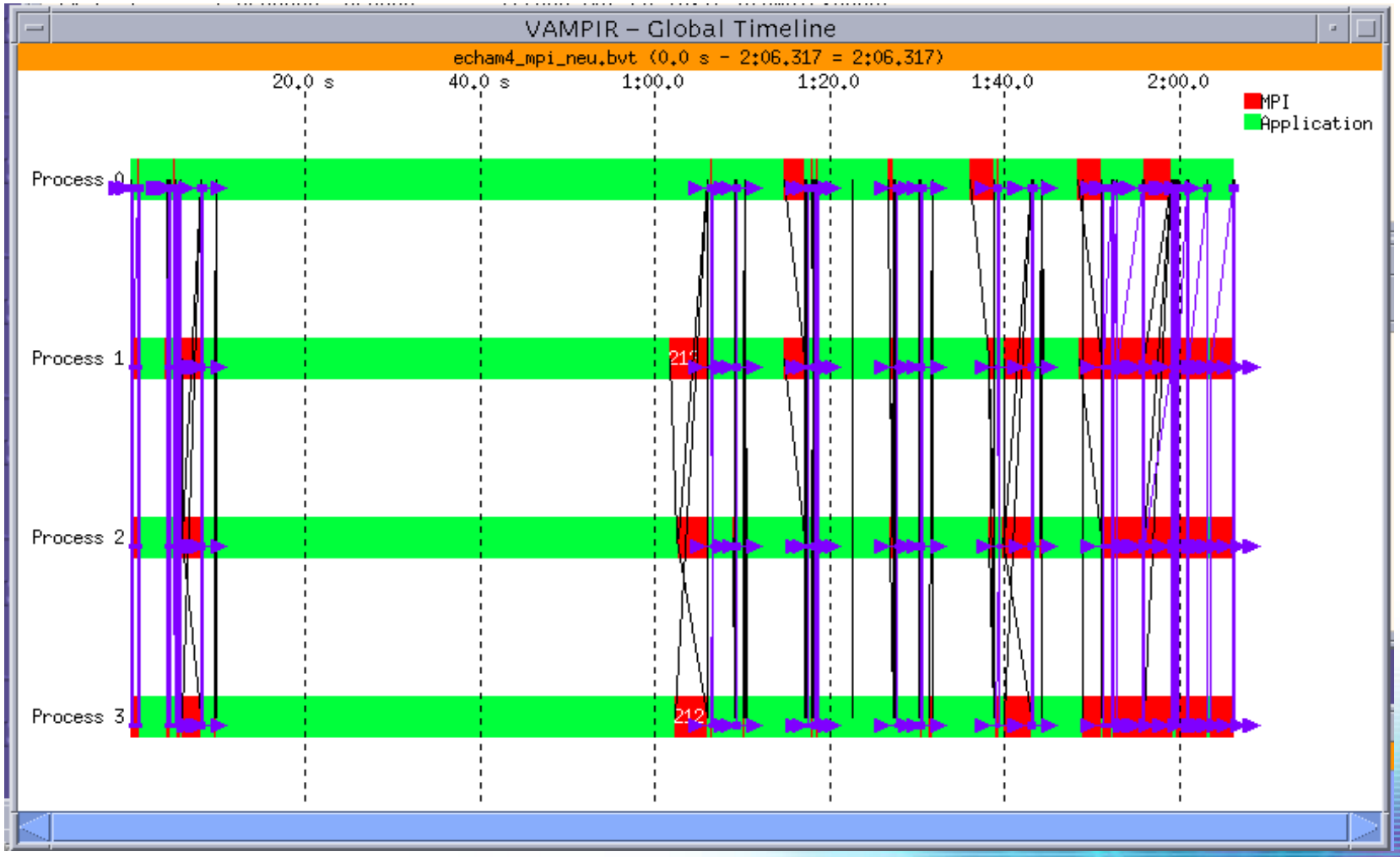
Debugging:

```
(dbx) cont
[2] stopped in MAIN_ at line 25 in file "myprog.f"
* 25   INT4=INT4*2
(dbx)

(dbx) quit
```

Linking MPI Programs with Vampirtrace/SX

```
sam% sxmpif90 -vampir mpi_1.f
mpi_1.f:
sam%


cs1 % mpirun -np 2 a.out
 number of processors:   2
 MPI_comm_rank(  1  ): born to be alive
 number of processors:   2
 MPI_comm_rank(  0  ): born to be alive
Writing tracefile a.out.bvt
cs1 %
```

Global Timeline for all processes of ECHAM 4.6 (4 processor run): Time in seconds



The End

For questions after the course:

mgalle@hpce.nec.com
sborowski@hpce.nec.com