SR8000 – Performance Profiling and Tuning

Profiling	C	**Flags:** Profiling –Xmonitor –Wb,Phopt,monitor_loop=FALSE **Flags:** Compiler log –loglist **Linking:** –lpl
	Fortran	**Flags:** Profiling –Xparmonitor **Flags:** Compiler log –loglist **Linking:** –lpl

Always use these flags when developing code. It helps.

– foo.log

```
*poption parallel
 ** Parallel processing starting at loop entry
 ** Parallel function: _parallel_func_1_DDOT
 ** Parallel loop
 **    S: reduction variable (SUM)
 ** --- Add barrier at loop exit ---
 ** Parallel processing finishing at loop exit
 **
 **    Innermost loop accumulator variables expanded (8 times).
 **    PVP applied
       do i = 1,n
         s = s + a(i)*b(i)
       enddo
       end
```

debis Systemhaus Solutions for Research

Critical log entries **compas related**	Loop not parallelized	Check your code. There might be loop dependencies you were not aware of. If there aren't any problems: Help the compiler via compiler directives. The most useful ones are *poption tlocal *poption parallel *poption indep *poption init *poption lasvf

Critical log entries **pvp related**	Prefetch not applied Preload not applied SWPL not applied PVP not applied	Again: Check the code. Help the compiler via compiler directives. The most useful ones are *soption unroll(n) *soption predicate *voption indep(a) *voption prefetch(a) *voption preload(a) *voption nopreload(a) *voption speculative

Agressive compile flags	C	**Flags:** –64 –C –Os +Op –predicate +DB3 –(no)parallel –pvfunc=3 –(no)scope –(no)strideprefetch –restrict –(no)preload –Ouinline=extended –Oinlinemsg –nolimit For details please check the manual. **Note:** C supports only a handful of compiler directives. The only way to resolve problems therefore are changes in the code or compile options. –noscope –restrict and –nostridepretch are the most useful ones in that respect. **Warning: +**Op guarantees non aliased parameter list.

Agressive compile flags	Fortran	**Flags:** –64 –Oss –pvfunc=3 –(no)parallel **Note:** In fortran the best way to resolve problems is via compiler directives.

Examples	Fdtd – Finite difference time domain snippet
	Dgemmtt – Transposed matrix multiply
	Dcopy – Parallel copy
	We'll provide several examples of fortran and c–code in the practical part of the course. The examples will be made available on www.hlrs.de before the course.

– tdtd.F

```
      do k=2, NK-1
*soption unroll(2)
         do j=2, NJ-1
*soption unroll(2)
            do i=2, NI-1
               Ex(i,j,k) =
     $               0.5*(EPSINV(i,j,k) +  EPSINV(i+1,j,k))*
     $               (dty*( Hz(i,j,k)- Hz(i,j-1,k))-
     $               dtz*( Hy(i,j,k)- Hy(i,j,k-1)))+ Ex(i,j,k)
               Ey(i,j,k) =
     $               0.5*(EPSINV(i,j,k) +  EPSINV(i,j+1,k))*
     $               (dtz*( Hx(i,j,k)- Hx(i,j,k-1))-
     $               dtx*( Hz(i,j,k)- Hz(i-1,j,k)))+ Ey(i,j,k)
               Ez(i,j,k) =
     $               0.5*(EPSINV(i,j,k) +  EPSINV(i,j,k+1))*
     $               (dtx*( Hy(i,j,k)- Hy(i-1,j,k))-
     $               dty*( Hx(i,j,k)- Hx(i,j-1,k)))+ Ez(i,j,k)

            enddo
         enddo
      enddo
```


– dgemmtt.F

```
*soption unroll(1)
      do  j = 1, n, 8
*soption unroll(8)
         do  i = 1, m
            temp1 = zero
                   ....
            temp8 = zero

*soption unroll(1)
            do  l = 1, k
               temp1 = temp1 + a( l, i )*b( j+0, l )
                       ....
               temp8 = temp8 + a( l, i )*b( j+7, l )
            enddo
            c( i, j ) = alpha*
     $           (temp1+temp2+temp3+temp4
     $           +temp5+temp6+temp7+temp8)
     $           + beta*c( i, j )
         enddo
      enddo
```


– dcopy.F

```
      if ((incx.eq.1).and.(incy.eq.1)) goto hell
        ix = 1
        iy = 1
         do 10 i = 1,n
           zy(iy) = zx(ix)
           ix = ix + incx
           iy = iy + incy
      10 continue



      if ((incx.eq.1).and.(incy.ne.1)) then
*poption parallel
          do i = 0,n-1
             zy(i*incy+1) = zx(i+1)
          enddo
      endif
```