```
do jlev = 1,nlay ! Start at top-of-atmosphere
 nreg = nregions
 if (is_clear_sky_layer(jlev) nreg = 1

 do jreg = 1,nreg ! Loop over relevant regions (only 1 if layer is clear-sky)
  if (jreg == 1) then ! optical properties are equal to clear-sky values
   optical_depth_tot  =  optical_depth(:,jlev,jcol)
   ssa_tot = ssa(:,jlev,jcol)
   g_tot   =   g(:,jlev,jcol)
  else
   do jg = 1,ng ! loop over g-points
    ! Cloudy-sky optical properties from band-wise cloud values and g-point-wise clear-sky values
    optical_depth_tot(jg) =  optical_depth(jg,jlev,jcol) + ...
    ...
   end do
  end if

  call calc_two_stream_gammas_sw(ng,  mu0, ssa_tot, g_tot,  gamma1, gamma2, gamma3)
  call calc_reftrans_sw(ng, mu0, optical_depth_tot, ssa_tot, gamma1, gamma2, gamma3, &
   &  reflectance(:,jreg,jlev), transmittance(:,jreg,jlev), & ! outputs
   &  ref_dir(:,jreg,jlev), trans_dir_diff(:,jreg,jlev), trans_dir_dir(:,jreg,jlev)) ! outputs
 end do
end do
```

```
! Computations for clear-sky region as a separate step: collapse the two inner dimensions
call calc_reftrans_sw_opt(ng*nlay, mu0, optical_depth(:,:,jcol), ssa(:,:,jcol), g(:,:,jcol), &
 &  reflectance_clear, transmittance_clear, ref_dir_clear, trans_dir_diff_clear, trans_dir_dir_clear)

! Cloudy computations: start at top-of-atmosphere and find first cloudy layer, if one exists
any_clouds_below = .false.
jtop = findloc(is_clear_sky_layer(1:nlay), .false., dim=1)
if (jtop>0) any_clouds_below = .true.

do while (any_clouds_below)
 ! Find the bottom of this cloud
 jbot = ...
 nlay_cloud = jbot - jtop + 1
 allocate(optical_depth_tot_cloudy(ng,2:nreg,jtop:jbot), ssa_tot_cloudy(ng,2:nreg,jtop:jbot), &
    & g_tot_cloudy(ng,2:nreg,jtop:jbot))

 do jlev = jtop, jbot
  do jreg = 2, nregions ! = 3
   do jg = 1,ng
    ! Spectral cloudy-sky optical properties from band-wise cloud values and spectral clear-sky values
    optical_depth_tot_cloudy(jg,jreg,jlev) = ...
    ...
   end do
  end do
 end do

 call calc_reftrans_sw_opt(ng*2*nlay_cloud, & ! g-points * cloudy regions * adjacent cloudy layers
  &  mu0, optical_depth_tot_cloudy, ssa_tot_cloudy, g_tot_cloudy, &
  &  reflectance(:,:,jtop:jbot), transmittance(:,:,jtop:jbot), & ! outputs
  &  ref_dir(:,:,jtop:jbot), trans_dir_diff(:,:,jtop:jbot), trans_dir_dir(:,:,jtop:jbot)) ! outputs

 deallocate(optical_depth_tot_cloudy, ssa_tot_cloudy, g_tot_cloudy)

 ! Does another cloudy layer exist? If not, set logical to false to exit "while"
 if (jbot== nlay) any_clouds_below=.false. ! surface reached

 if (any(.not. is_clear_sky_layer(jbot+1:nlay))) then
  ! find the top of the new cloud
  jtop = ...
 else
  any_clouds_below=.false.
 end if
end do
```

Figure 1: Refactoring of TripleClouds-SW. In addition to optimizing and fusing kernels, in the new code (bottom) the reflectance-transmittance computations are performed in a batched manner for multiple layers by collapsing the spectral and vertical dimensions.