Packing the buffers on the accelerator

```
!$acc data present(grad,recv_w,recv_e,send_e,send_w,recv_n,&
!$acc&                  recv_s,send_n,send_s)
!$acc parallel_loop
      do k=-1,lz
        do j=-1,local_ly
          send_e(j,k) = grad(local_lx-1,j        ,k)
          send_w(j,k) = grad(0         ,j        ,k)
        end do
      end do
!$acc end parallel_loop
!$acc update host(send_e,send_w)
      call mpi_irecv(recv_w, bufsize(2),mpi_double_precision,w_id, &
           tag(25),mpi_comm_world,irequest_in(25),ierr)
          o  o  o
      call mpi_isend(send_w, bufsize(2),mpi_double_precision,w_id, &
           tag(26),& mpi_comm_world,irequest_out(26),ierr)
      call mpi_waitall(2,irequest_in(25),istatus_req,ierr)
      call mpi_waitall(2,irequest_out(25),istatus_req,ierr)
!$acc update device(recv_e,recv_w)
!$acc parallel
!$acc loop
      do k=-1,lz
        do j=-1,local_ly
          grad(local_lx  ,j        ,k) = recv_e(j,k)
          grad(-1        ,j        ,k) = recv_w(j,k)
```

Final Profile - bulk of time in kernel execution

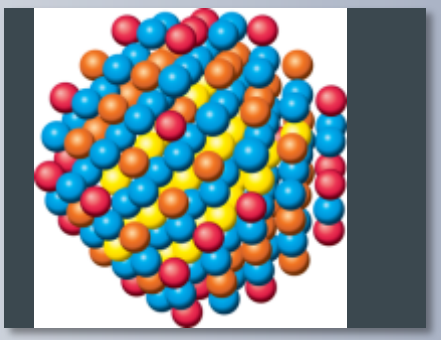
```
 |  37.9% | 236.592782 |        -- |     -- | 11403.0 |ACCELERATOR
||-----------------------------------------------------------------------
||  15.7% |  98.021619 | 43.078137 |  31.0% |   200.0 |lbm3d2p_d_.ASYNC_KERNEL@li.129
||   3.7% |  23.359080 |  2.072147 |   8.3% |   200.0 |lbm3d2p_d_.ASYNC_KERNEL@li.127
||   3.6% |  22.326085 |  1.469419 |   6.3% |   200.0 |lbm3d2p_d_.ASYNC_KERNEL@li.132
||   3.0% |  19.035232 |  1.464608 |   7.3% |   200.0 |collisionb_.ASYNC_KERNEL@li.599
||   2.6% |  16.216648 |  3.505232 |  18.1% |   200.0 |lbm3d2p_d_.ASYNC_KERNEL@li.131
||   2.5% |  15.401916 |  8.093716 |  35.0% |   200.0 |injection_.ASYNC_KERNEL@li.1116
||   1.9% |  11.734026 |  4.488785 |  28.1% |   200.0 |recolor_.ASYNC_KERNEL@li.786
||   0.9% |   5.530201 |  2.132243 |  28.3% |   200.0 |collisionb_.SYNC_COPY@li.593
||   0.8% |   4.714995 |  0.518495 |  10.1% |   200.0 |collisionb_.SYNC_COPY@li.596
||   0.6% |   3.738615 |  2.986891 |  45.1% |   200.0 |collisionb_.ASYNC_KERNEL@li.568
||   0.4% |   2.656962 |  0.454093 |  14.8% |     1.0 |lbm3d2p_d_.ASYNC_COPY@li.100
||   0.4% |   2.489231 |  2.409892 |  50.0% |   200.0 |streaming_exchange_.ASYNC_COPY@li.810
||   0.4% |   2.487132 |  2.311190 |  48.9% |   200.0 |streaming_exchange_.ASYNC_COPY@li.625
||   0.2% |   1.322791 |  0.510645 |  28.3% |   200.0 |streaming_exchange_.SYNC_COPY@li.622
||   0.2% |   1.273771 |  0.288743 |  18.8% |   200.0 |streaming_exchange_.SYNC_COPY@li.574
||   0.2% |   1.212260 |  0.298053 |  20.0% |   200.0 |streaming_exchange_.SYNC_COPY@li.759
||   0.2% |   1.208250 |  0.422182 |  26.3% |   200.0 |streaming_exchange_.SYNC_COPY@li.806
||   0.1% |   0.696120 |  0.442372 |  39.5% |   200.0 |streaming_exchange_.ASYNC_KERNEL@li.625
||   0.1% |   0.624982 |  0.379697 |  38.4% |   200.0 |streaming_exchange_.ASYNC_KERNEL@li.525
```

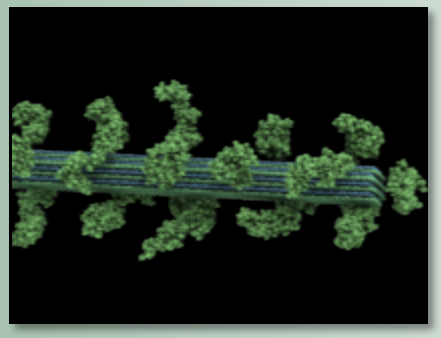
Cray GPU Programming Environment

- Objective: Enhance productivity related to porting applications to hybrid multi-core systems
- Four core components
  - Cray Statistics Gathering Facility on host and GPU
  - Cray Optimization Explorer – Scoping Tools (COE)
  - Cray Compilation Environment (CCE)
  - Cray GPU Libraries

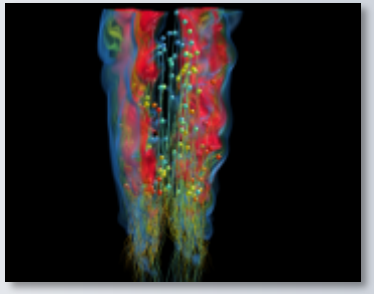
Titan: Early Science Applications

WL-LSMS
Role of material disorder, statistics, and fluctuations in nanoscale materials and systems.



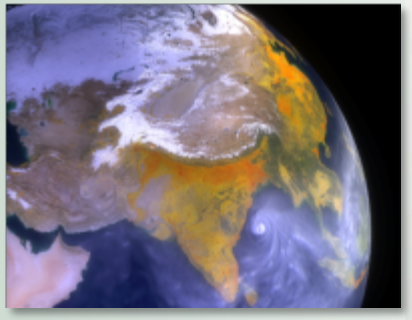


LAMMPS
Biofuels: An atomistic model of cellulose (blue) surrounded by lignin molecules comprising a total of 3.3 million atoms. Water not shown.

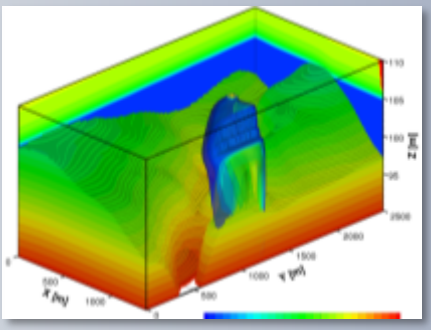


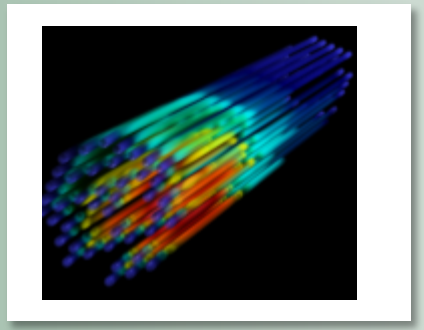
S3D
How are going to efficiently burn next generation diesel/bio fuels?
.

CAM / HOMME
Answer questions about specific climate change adaptation and mitigation scenarios; realistically represent features like precipitation patterns/statistics and tropical storms



PFLOTRAN
Stability and viability of large scale CO_2 sequestration; predictive containment groundwater transport

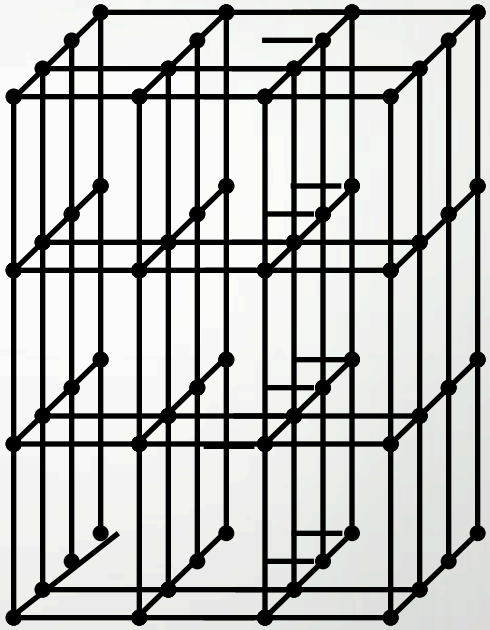




Denovo
Unprecedented high-fidelity radiation transport calculations that can be used in a variety of nuclear energy and technology applications.

S3D – A DNS solver

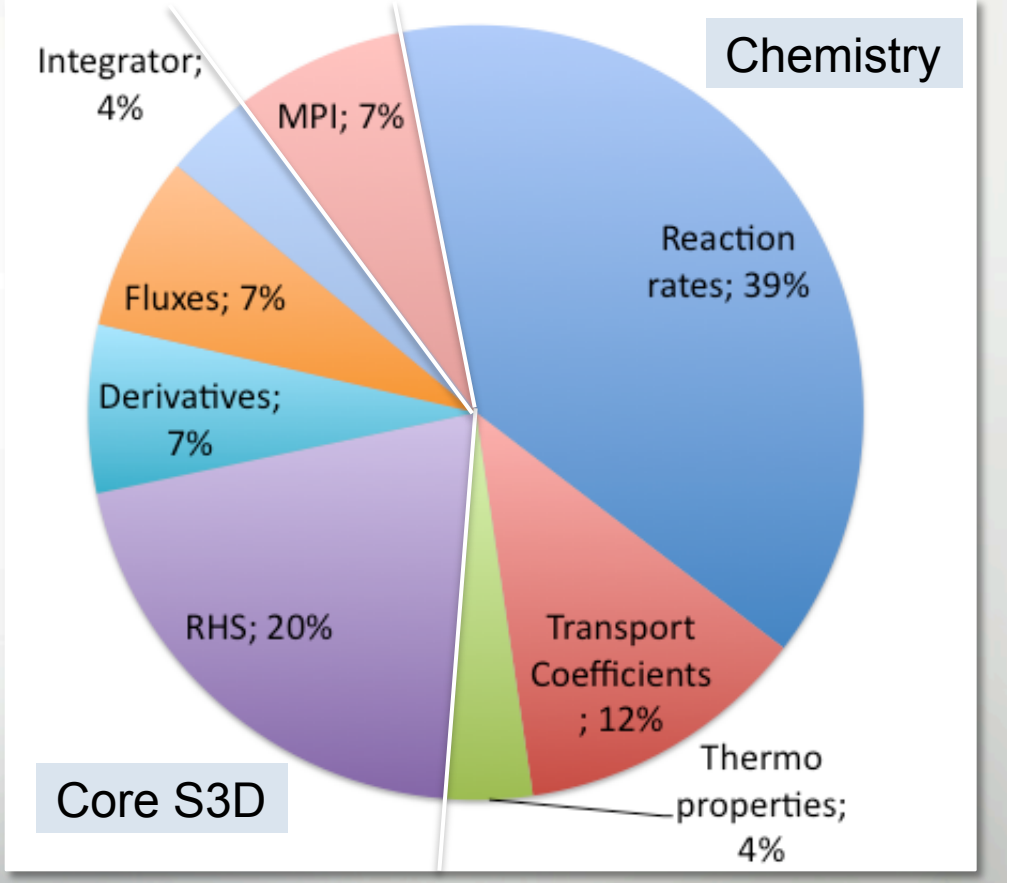
- Structured Cartesian mesh flow solver
- Solves compressible reacting Navier-Stokes, energy and species conservation equations.
  - 8th order explicit finite difference method
  - 4th order Runge-Kutta integrator with error estimator
- Detailed gas-phase thermodynamic, chemistry and molecular transport property evaluations
- Lagrangian particle tracking
- MPI-1 based spatial decomposition and parallelism
- Fortran code. Does not need linear algebra, FFT or solver libraries.



Developed and maintained at CRF, Sandia (Livermore) with BES and ASCR sponsorship. PI – Jacqueline H. Chen (jhchen@sandia.gov)

Benchmark Problem and Profile

- A benchmark problem was defined to closely resemble the target simulation
  - 52 species n-heptane chemistry and 48^3 grid points per node
    - 48^3 * 18,500 nodes = 2 billion grid points
    - Target problem would take two months on today's Jaguar
- Code was benchmarked and profiled on dual-hex core XT5
- Several kernels identified and extracted into stand-alone driver programs



Acceleration Strategy

Team:

Ramanan Sankaran	ORNL
Ray Grout	NREL
John Levesque	Cray

Goals:

- Convert S3D to a hybrid multi-core application suited for a multi-core node with or without an accelerator.
- Be able to perform the computation entirely on the accelerator.
  - Arrays and data able to reside entirely on the accelerator.
  - Data sent from accelerator to host CPU for halo communication, I/O and monitoring only.

Strategy:

- To program using both hand-written and generated code.
  - Hand-written and tuned CUDA*.
  - Automated Fortran and CUDA generation for chemistry kernels
  - Automated code generation through compiler directives
- S3D is now a part of Cray's compiler development test cases

Original S3D

		S3D		
Time Step		Solve_Drive		
Time Step	Runge K	Integrate		
Time Step	Runge K	RHS		
Time Step	Runge K		get mass fraction	I,j,k,n_spec loops
Time Step	Runge K		get_velocity	I,j,k,n_spec loops
Time Step	Runge K		calc_inv_avg	I,j,k,n_spec loops
Time Step	Runge K		calc_temp	I,j,k,n_spec loops
Time Step	Runge K		Compute Grads	I,j,k,n_spec loops
Time Step	Runge K		Diffusive Flux	I,j,k,n_spec loops
Time Step	Runge K		Derivatives	I,j,k,n_spec loops
Time Step	Runge K		reaction rates	I,j,k,n_spec loops

Profile from Original S3D

Table 1: Profile by Function Group and Function

Time%	Time	Imb. Time	Imb. Time%	Calls	Group / Function / PE=HIDE / Thread=HIDE
100.0%	284.732812	--	--	156348682.1	Total
92.1%	262.380782	--	--	155578796.1	USER
12.4%	35.256420	0.237873	0.7%	391200.0	ratt_i_.LOOPS
9.6%	27.354247	0.186752	0.7%	391200.0	ratx_i_.LOOPS
7.7%	21.911069	1.037701	4.5%	1562500.0	mcedif_.LOOPS
5.4%	15.247551	2.389440	13.6%	35937500.0	mceval4_
5.2%	14.908749	4.123319	21.7%	600.0	rhsf_.LOOPS
4.7%	13.495568	1.229034	8.4%	35937500.0	mceval4_.LOOPS
4.6%	12.985353	0.620839	4.6%	701.0	calc_temp$thermchem_m_.LOOPS
4.3%	12.274200	0.167054	1.3%	1562500.0	mcavis_new$transport_m_.LOOPS
4.0%	11.363281	0.606625	5.1%	600.0	computespeciesdiffflux$transport_m_.LOOPS
2.9%	8.257434	0.743004	8.3%	21921875.0	mixcp$thermchem_m_
2.9%	8.150646	0.205423	2.5%	100.0	integrate_.LOOPS
2.4%	6.942384	0.078555	1.1%	391200.0	qssa_i_.LOOPS
2.3%	6.430820	0.481475	7.0%	21921875.0	mixcp$thermchem_m_.LOOPS
2.0%	5.588500	0.343099	5.8%	600.0	computeheatflux$transport_m_.LOOPS
1.8%	5.252285	0.062576	1.2%	391200.0	rdwdot_i_.LOOPS
1.7%	4.801062	0.723213	13.1%	31800.0	derivative_x_calc_.LOOPS
1.6%	4.461274	1.310813	22.7%	31800.0	derivative_y_calc_.LOOPS
1.5%	4.327627	1.290121	23.0%	31800.0	derivative_z_calc_.LOOPS
1.4%	3.963951	0.138844	3.4%	701.0	get_mass_frac$variables_m_.LOOPS

Restructured S3D for multi-core systems

		S3D	
Time Step		Solve_Drive	
Time Step	Runge K	Integrate	
Time Step	Runge K	RHS	
Time Step	Runge K	grid loop -omp	get mass fraction
Time Step	Runge K	grid loop-omp	get_velocity
Time Step	Runge K	grid loop-omp	calc_inv_avg
Time Step	Runge K	grid loop-omp	calc_temp
Time Step	Runge K	grid loop-omp	Compute Grads
Time Step	Runge K	grid loop-omp	Diffusive Flux
Time Step	Runge K	grid loop-omp	Derivatives
Time Step	Runge K	grid loop-omp	reaction rates

Statistics from running S3D

Table 1: Profile by Function Group and Function

```
   Time%   |       Time   |     Imb.   |  Imb.   |     Calls   |Group
           |              |     Time   | Time%   |             | Function|------------------------
    85.3% | 539.077983 |        -- |      -- |  144908.0 |USER
||-------------------------------------------------------------------------------------------------
-------------
||  21.7% | 136.950871 | 0.583731 |   0.5% |      600.0 |rhsf_
||  14.7% |  93.237279 | 0.132829 |   0.2% |      600.0 |rhsf_.LOOP@li.1084
||   8.7% |  55.047054 | 0.309278 |   0.6% |      600.0 |rhsf_.LOOP@li.1098
||   6.3% |  40.129463 | 0.265153 |   0.8% |      100.0 |integrate_
||   5.8% |  36.647080 | 0.237180 |   0.7% |      600.0 |rhsf_.LOOP@li.1211
||   5.6% |  35.264114 | 0.091537 |   0.3% |      600.0 |rhsf_.LOOP@li.194
||   3.7% |  23.624271 | 0.054666 |   0.3% |      600.0 |rhsf_.LOOP@li.320
||   2.7% |  17.211435 | 0.095793 |   0.6% |      600.0 |rhsf_.LOOP@li.540
||   2.4% |  15.471160 | 0.358690 |   2.6% |    14400.0 |derivative_y_calc_buff_r_.LOOP@li.1784
||   2.4% |  15.113374 | 1.020242 |   7.2% |    14400.0 |derivative_z_calc_buff_r_.LOOP@li.1822
||   2.3% |  14.335142 | 0.144579 |   1.1% |    14400.0 |derivative_x_calc_buff_r_.LOOP@li.1794
||   1.9% |  11.794965 | 0.073742 |   0.7% |      600.0 |integrate_.LOOP@li.96
||   1.7% |  10.747430 | 0.063508 |   0.7% |      600.0 |computespeciesdiffflux2$transport_m_.LOOP
||   1.5% |   9.733830 | 0.096476 |   1.1% |      600.0 |rhsf_.LOOP@li.247
||   1.2% |   7.649953 | 0.043920 |   0.7% |      600.0 |rhsf_.LOOP@li.274
||   0.8% |   5.116578 | 0.008031 |   0.2% |      600.0 |rhsf_.LOOP@li.398
||   0.6% |   3.966540 | 0.089513 |   2.5% |        1.0 |s3d_
||   0.3% |   2.027255 | 0.017375 |   1.0% |      100.0 |integrate_.LOOP@li.73
||   0.2% |   1.318550 | 0.001374 |   0.1% |      600.0 |rhsf_.LOOP@li.376
||   0.2% |   0.986124 | 0.017854 |   2.0% |      600.0 |rhsf_.REGION@li.1096
||   0.1% |   0.700156 | 0.027669 |   4.3% |        1.0 |exit
```

Advantage of raising loops

- Create good granularity OpenMP Loop
- Improves cache re-use
- Reduces Memory usage significantly
- Creates a good potential kernel for an accelerator

Restructured S3D for multi-core systems

		S3D	
Time Step – acc_data		Solve_Drive	
Time Step– acc_data	Runge K	Integrate	
Time Step– acc_data	Runge K	RHS	
Time Step– acc_data	Runge K	grid loop -ACC	get mass fraction
Time Step– acc_data	Runge K	grid loop-ACC	get_velocity
Time Step– acc_data	Runge K	grid loop-ACC	calc_inv_avg
Time Step– acc_data	Runge K	grid loop-ACC	calc_temp
Time Step– acc_data	Runge K	grid loop-ACC	Compute Grads
Time Step– acc_data	Runge K	grid loop-ACC	Diffusive Flux
Time Step– acc_data	Runge K	grid loop-ACC	Derivatives
Time Step– acc_data	Runge K	grid loop-ACC	reaction rates

What does OpenACC look like

```
#ifdef GPU
!$acc data copyin(avmolwt, cpCoef_aa, cpCoef_bb, cpmix, enthCoef_aa, enthCoef_bb, &
!$acc& gamma, invEnthInc, iorder, lrmcwrk, mixMW, molwt_c, molwt, n_spec,neighbor, nsc, pressure,&
!$acc&  neg_f_x_buf, neg_f_y_buf, neg_f_z_buf, pos_f_x_buf, pos_f_y_buf, pos_f_z_buf, &
!$acc&  neg_fs_x_buf, neg_fs_y_buf, neg_fs_z_buf, pos_fs_x_buf, pos_fs_y_buf, pos_fs_z_buf, &
!$acc&  rk_alpha, rk_beta, rk_err, rmcwrk, Ru, temp, temp_hibound, temp_lobound, tstep, &
!$acc& u, vary_in_x, vary_in_y, vary_in_z, volum, yspecies,q,q_err)
#endif
```

What does OpenACC look like

```
#ifdef GPU
!$acc data present_or_create( avmolwt, cpcoef_aa, cpcoef_bb, cpmix, enthcoef_aa, enthcoef_bb, &
!$acc&     gamma, invEnthInc, lrmcwrk,  molwt_c, molwt, n_spec, pressure, q, neighbor, nsc, &
!$acc&     rhs, rmcwrk, Ru, temp, temp_hibound, temp_lobound, u, vary_in_x, vary_in_y, &
!$acc&     vary_in_z, volum, yspecies, ds_mxvg,  diffflux,tmmp2n,sumf1,sumf2,&
!$acc&   diffusion, grad_mixmw, grad_t, grad_u, grad_ys, h_spec,  lambdax, &
!$acc&    rr_r, rs_therm_diff,  tmmp, tmmp2, tmmpdx,voltmp, vscsty,&
!$acc&   neg_fs_x_buf, neg_fs_y_buf,neg_fs_z_buf, pos_fs_x_buf, pos_fs_y_buf, pos_fs_z_buf, &
!$acc$   buffer41,buffer42,buffer43,buffer44,buffer45, &
!$acc&   buffer31,buffer32,buffer33,buffer34,buffer35,buffer36,buffer37, &
!$acc&     neg_f_x_buf, neg_f_y_buf,neg_f_z_buf, pos_f_x_buf, pos_f_y_buf, pos_f_z_buf,mixmw)&
!$acc&    copyin( jstage,scale1x,scale1y,scale1z,aex,bex,cex,dex,ds,aey,bey,cey,dey,aez,bez,cez,dez)
#endif
```

What does OpenACC look like

```
#ifdef GPU
!$acc parallel loop  gang  private(i,ml,mu)
#else
!$omp parallel private(i, ml, mu)
!$omp do
#endif
 do i = 1, nx*ny*nz, ms
  ml = i
  mu =  min(i+ms-1, nx*ny*nz)
  call calc_gamma_r( gamma, cpmix, avmolwt, ml, mu)
  call calc_press_r( pressure, q(1,1,1,4), temp, avmolwt, ml, mu )
  call calc_specEnth_allpts_r(temp, h_spec, ml, mu)
end do
#ifdef GPU
!$acc end parallel loop
#else
!$omp end parallel
#endif
```

What OpenACC looks like

```
#ifdef GPU
!$acc parallel loop gang collapse(2) private(n,i,j,k)
#else
!$omp parallel do private(n,i,j,k)
#endif
  do n=1,n_spec
    do k = 1,nz
#ifdef GPU
!$acc loop vector collapse(2)
#endif
      do j = 1,ny
        do i = 1,nx
          grad_Ys(i,j,k,n,1) = 0.0
          grad_Ys(i,j,k,n,2) = 0.0
          grad_Ys(i,j,k,n,3) = 0.0
          if(i.gt.iorder/2 .and. i.le.nx-iorder/2) then
             grad_Ys(i,j,k,n,1) = scale1x(i)*(aex *( yspecies(i+1,j,k,n)-yspecies(i-1,j,k,n) )    &
                 + bex *( yspecies(i+2,j,k,n)-yspecies(i-2,j,k,n) )                              &
                 + cex *( yspecies(i+3,j,k,n)-yspecies(i-3,j,k,n) )                              &
                 + dex *( yspecies(i+4,j,k,n)-yspecies(i-4,j,k,n) ))
          endif
          if(j.gt.iorder/2 .and. j.le.ny-iorder/2) then
             grad_Ys(i,j,k,n,2) = scale1y(j)*(aey *( yspecies(i,j+1,k,n)-yspecies(i,j-1,k,n) )    &
                 + bey *( yspecies(i,j+2,k,n)-yspecies(i,j-2,k,n) )                              &
                 + cey *( yspecies(i,j+3,k,n)-yspecies(i,j-3,k,n) )                              &
                 + dey *( yspecies(i,j+4,k,n)-yspecies(i,j-4,k,n) ))
          endif
          if(k.gt.iorder/2 .and. k.le.nz-iorder/2) then
             grad_Ys(i,j,k,n,3) = scale1z(k)*(aez *( yspecies(i,j,k+1,n)-yspecies(i,j,k-1,n) ) &
                 + bez *( yspecies(i,j,k+2,n)-yspecies(i,j,k-2,n) ) &
                 + cez *( yspecies(i,j,k+3,n)-yspecies(i,j,k-3,n) ) &
                 + dez *( yspecies(i,j,k+4,n)-yspecies(i,j,k-4,n) ))
          endif
        end do  ! i
      end do    ! j
#ifdef GPU
!$acc end loop
#endif
    end do      ! k
  end do        ! n
#ifdef GPU
!$acc end parallel loop
#endif
```

What does OpenACC look like

```
#ifdef GPU
#ifdef GPU_STREAMS
!$acc update host(pos_fs_x_buf(:,:,:,idx)) async(istr)
#else
!$acc host_data use_device(pos_fs_x_buf)
#endif
#endif
#ifdef GPU_STREAMS
      call cray_mpif_isend_openacc(c_loc(pos_fs_x_buf(1,1,1,idx)),(my*mz*iorder/2),&
                     MPI_REAL8,deriv_x_list(idx)%neg_nbr,idx+deriv_list_size,&
                     gcomm,istr,deriv_x_list(idx)%req(2),ierr)
#else
      call MPI_ISend(pos_fs_x_buf(1,1,1,idx),(my*mz*iorder/2),&
                     MPI_REAL8,deriv_x_list(idx)%neg_nbr,idx+deriv_list_size,&
                     gcomm,deriv_x_list(idx)%req(2),ierr)
#endif
#ifdef GPU
#ifndef GPU_STREAMS
!$acc end host_data
#endif
#endif
```

What does OpenACC look like

```
#ifdef GPU
!$acc host_data use_device(neg_f_x_buf)
#endif
     call MPI_IRecv(neg_f_x_buf(1,1,1,idx),(my*mz*iorder/2),&
                    MPI_REAL8,deriv_x_list(idx)%neg_nbr,idx,&
                    gcomm,deriv_x_list(idx)%req(1),ierr)
#ifdef GPU
!$acc end host_data
#endif
  endif
  if(lnbr(2)>=0) then
   ! get ghost cells from neighbor on (+x) side
#ifdef GPU
!$acc host_data use_device(pos_f_x_buf)
#endif
     call MPI_IRecv(pos_f_x_buf(1,1,1,idx),(my*mz*iorder/2),&
                    MPI_REAL8,deriv_x_list(idx)%pos_nbr,idx+deriv_list_size,&
                    gcomm,deriv_x_list(idx)%req(3),ierr)
#ifdef GPU
!$acc end host_data
#endif
  endif
```

!$acc host_data use_device

```
#ifdef GPU
!$acc data present(f)
!$acc host_data use_device(f)
#endif
  if( deriv_z_list(idx)%packed ) then
    deriv_z_list(idx)%packed = .false.
    if(deriv_z_list(idx)%neg_nbr>=0) then
       call MPI_ISend(f(1,1,1),(mx*my*iorder/2),&
                      MPI_REAL8,deriv_z_list(idx)%neg_nbr,deriv_list_size + idx, &
                      gcomm,deriv_z_list(idx)%req(2),ierr)
    endif
    if(deriv_z_list(idx)%pos_nbr>=0) then
     ! send ghost cells to neighbor on (+z) side
       nm = mz + 1 - iorder/2
       call MPI_ISend(f(1,1,nm),(mx*my*iorder/2), &
                      MPI_REAL8,deriv_z_list(idx)%pos_nbr,idx, &
                      gcomm,deriv_z_list(idx)%req(4),ierr)
    endif
  else
    if(deriv_z_list(idx)%neg_nbr>=0) then
       call MPI_ISend(f(1,1,1),(mx*my*iorder/2),&
                      MPI_REAL8,deriv_z_list(idx)%neg_nbr,deriv_list_size + idx, &
                      gcomm,deriv_z_list(idx)%req(2),ierr)
    endif

    if(deriv_z_list(idx)%pos_nbr>=0) then
     ! send ghost cells to neighbor on (+z) side
       nm = mz + 1 - iorder/2
       call MPI_ISend(f(1,1,nm),(mx*my*iorder/2), &
                      MPI_REAL8,deriv_z_list(idx)%pos_nbr,idx, &
                      gcomm,deriv_z_list(idx)%req(4),ierr)
    endif
  endif
#ifdef GPU
!$acc end host_data
!$acc end data
#endif
```


Oak Ridge National Laboratory

Thank You!

CRAY
THE SUPERCOMPUTER COMPANY