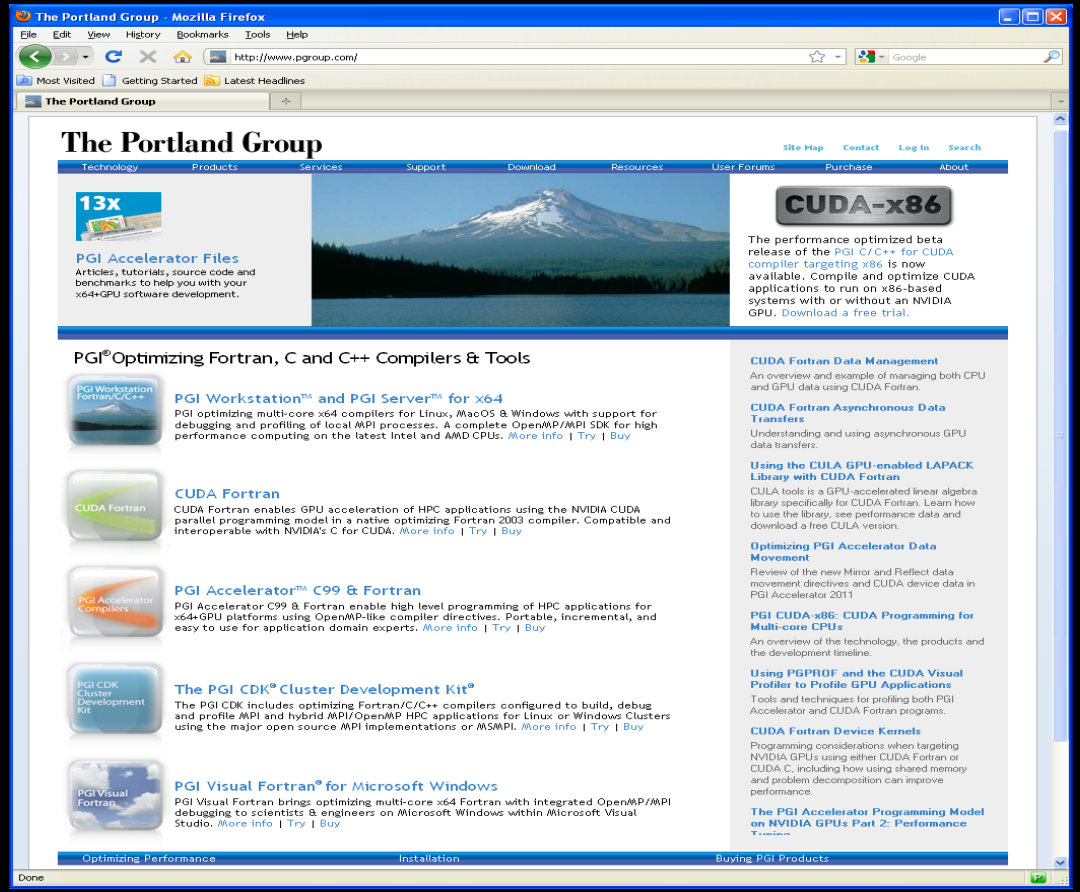
5x in 5 hours
Porting SEISMIC_CPML using the PGI Accelerator Model

PGI

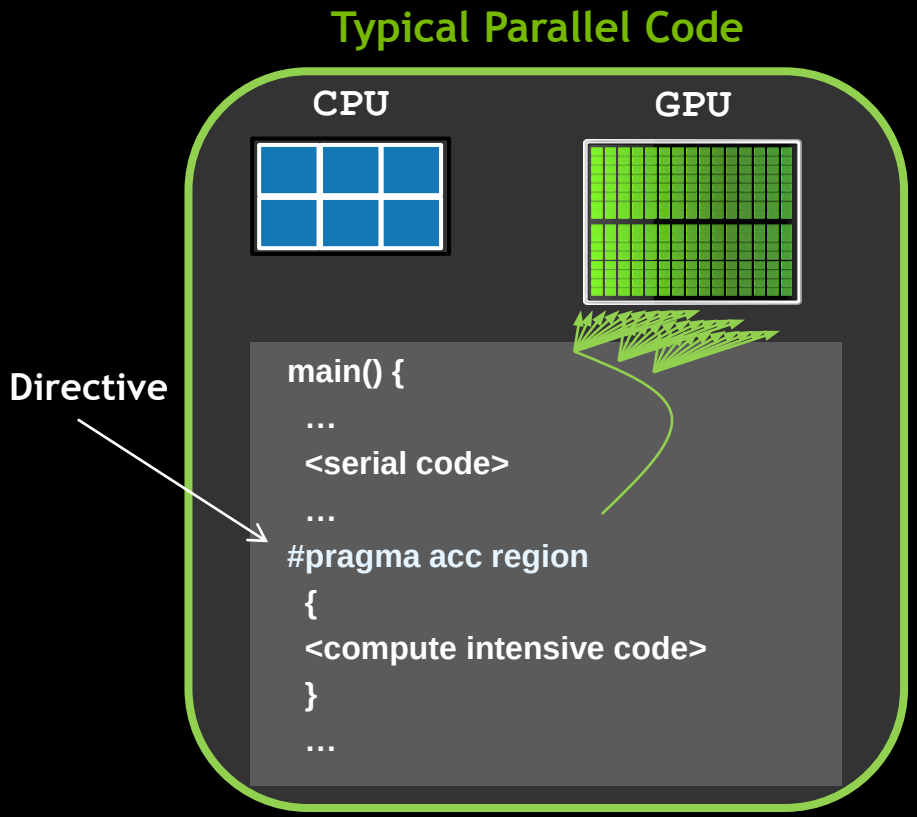
PGI

- C99, C++, F2003 Compilers
  - Optimizing
  - Vectorizing
  - Parallelizing
- Graphical parallel tools
  - PGDBG® debugger
  - PGPROF® profiler
- Intel, AMD, NVIDIA
- PGI Unified Binary™
- Linux, MacOS, Windows
- Visual Studio integration
- GPGPU Features
  - CUDA Fortran/C/C++
  - PGI Accelerator™
  - CUDA-x86



www.pgroup.com

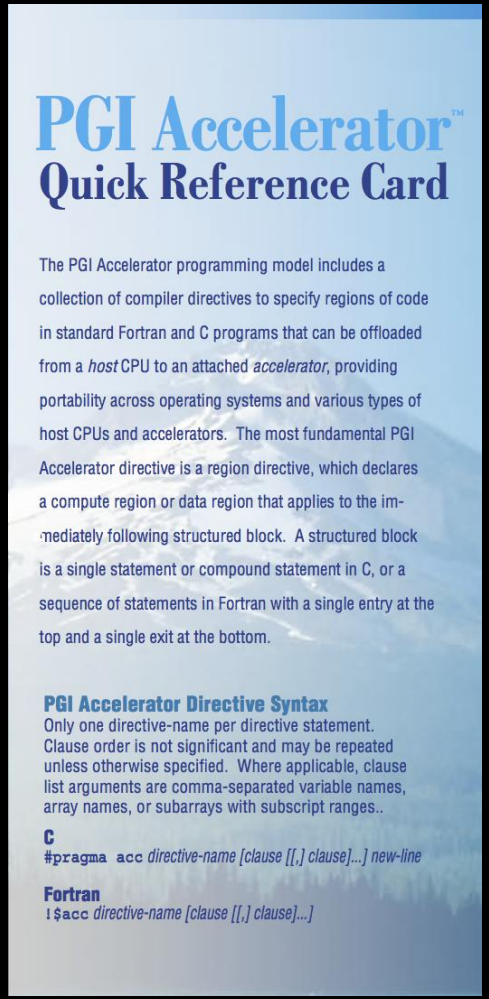
PGI Accelerator Directives



The quickest path to massively parallel Fortran or C applications for NVIDIA GPUs

Advantages?

Compiler Hints	Compiler handles all bookkeeping details
Productivity	Easy to get started, incremental
Portability	Single source tree and binary for CPUs + GPUs



PGI Accelerator™
Quick Reference Card

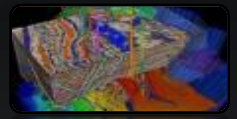
The PGI Accelerator programming model includes a collection of compiler directives to specify regions of code in standard Fortran and C programs that can be offloaded from a *host* CPU to an attached *accelerator*, providing portability across operating systems and various types of host CPUs and accelerators. The most fundamental PGI Accelerator directive is a region directive, which declares a compute region or data region that applies to the immediately following structured block. A structured block is a single statement or compound statement in C, or a sequence of statements in Fortran with a single entry at the top and a single exit at the bottom.

PGI Accelerator Directive Syntax
Only one directive-name per directive statement. Clause order is not significant and may be repeated unless otherwise specified. Where applicable, clause list arguments are comma-separated variable names, array names, or subarrays with subscript ranges..

C
`#pragma acc` *directive-name [clause [[,] clause]...] new-line*

Fortran
`!$acc` *directive-name [clause [[,] clause]...]*

Small Effort. Real Impact.



Large Oil Company

Dr. Jorge Pita

7 days and 3X

Solving billions of equations iteratively for oil exporation at world's largest petroleum reservoirs

UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

Univ. of Houston

Prof. Kayali

2 days and 20X

Analyzing magnetostatic interaction for innovations in areas such as storage, memories, and biosensing

THE UNIVERSITY OF MELBOURNE

Uni. Of Melbourne

Prof. Black

2 Days and 60x

Better understand complex reasons by lifecycles of snapper fish in Port Phillip Bay

УГАТУ

Ufa State Aviation

Prof. Arthur Yuldashev

4 Weeks and 7X

Generating stochastic geological models of oilfield reservoirs with borehole data

COMPUTING FOR SCIENCE
fs

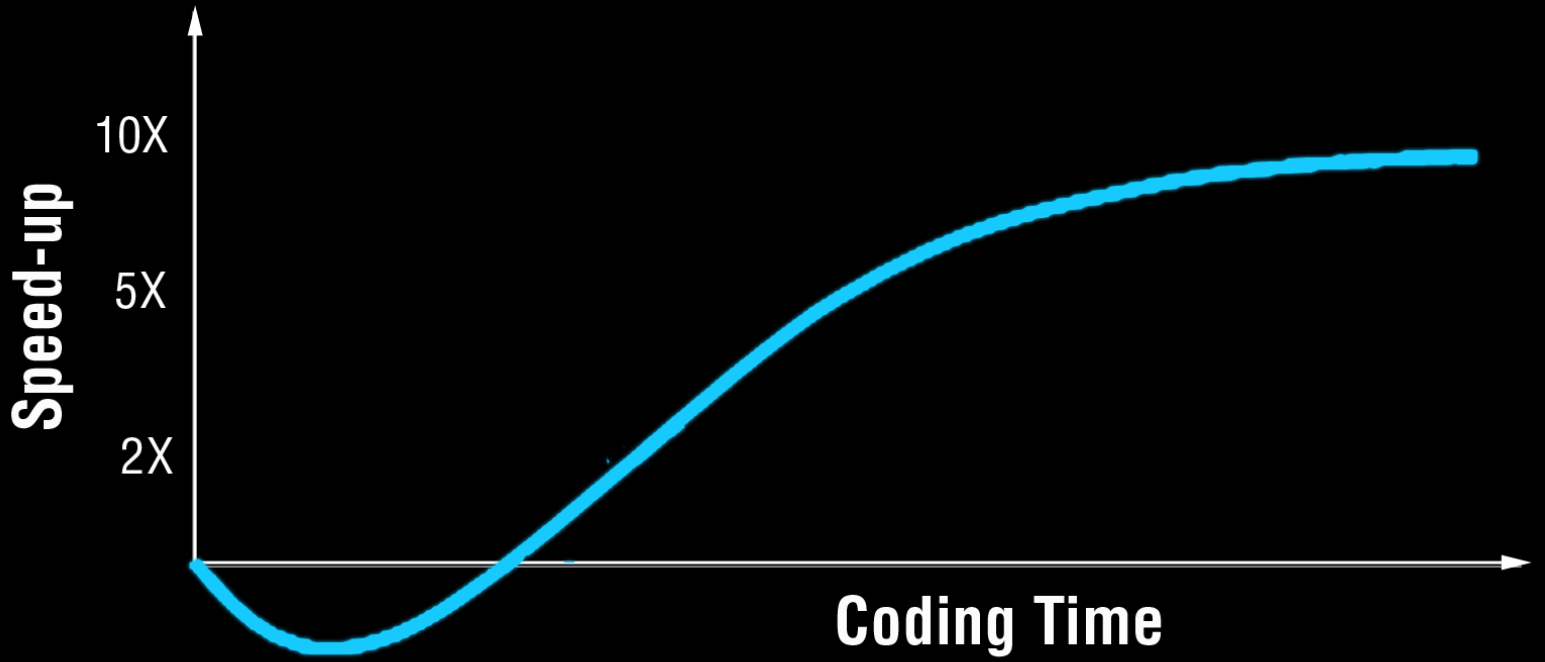
GAMESS-UK

Prof. Karl Wilkinson

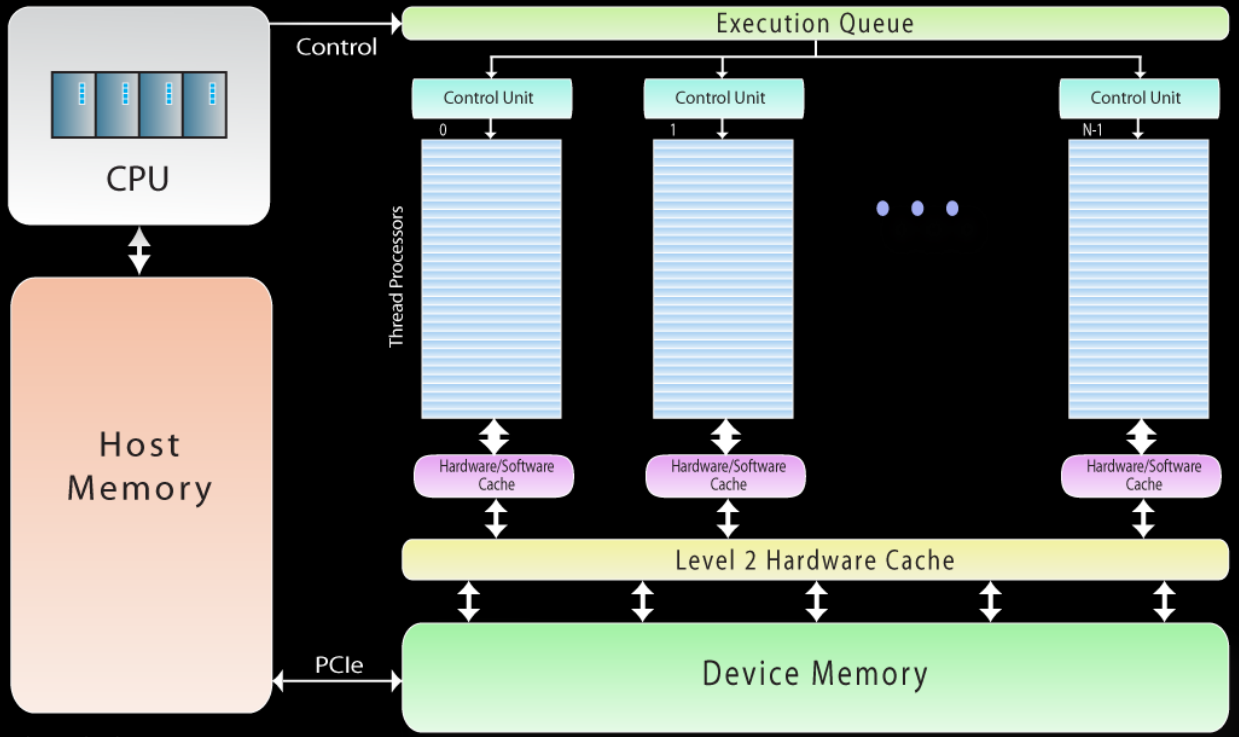
10x

Used for various fields such as investigating biofuel production and molecular sensors.

Typical porting experience with PGI Accelerator directives



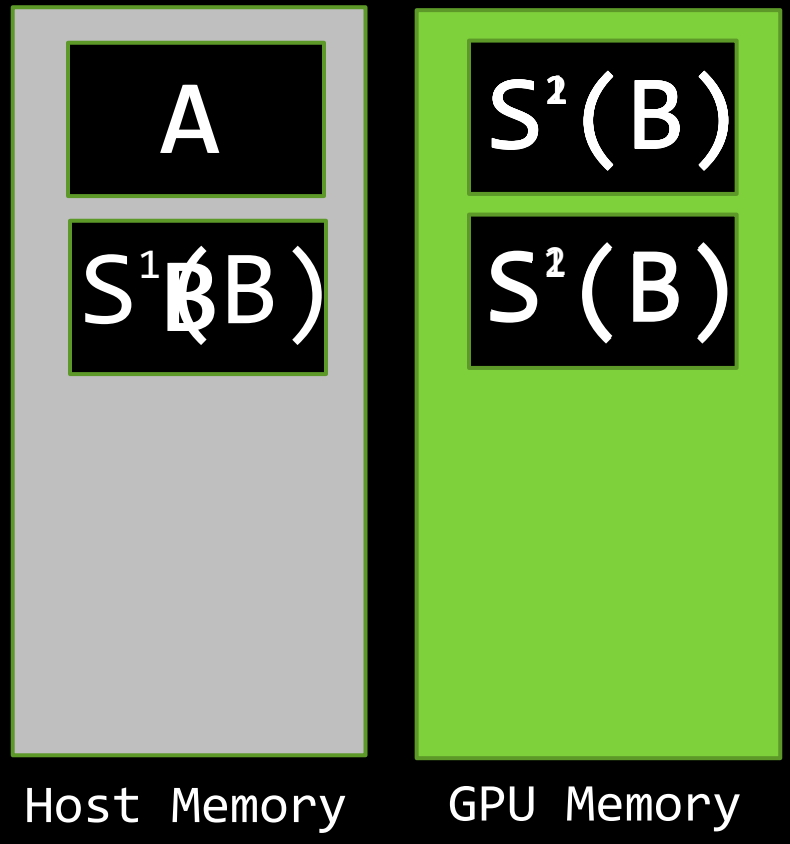
Multi-core x64 + NVIDIA GPU Architecture



Let’s look at a simple example …

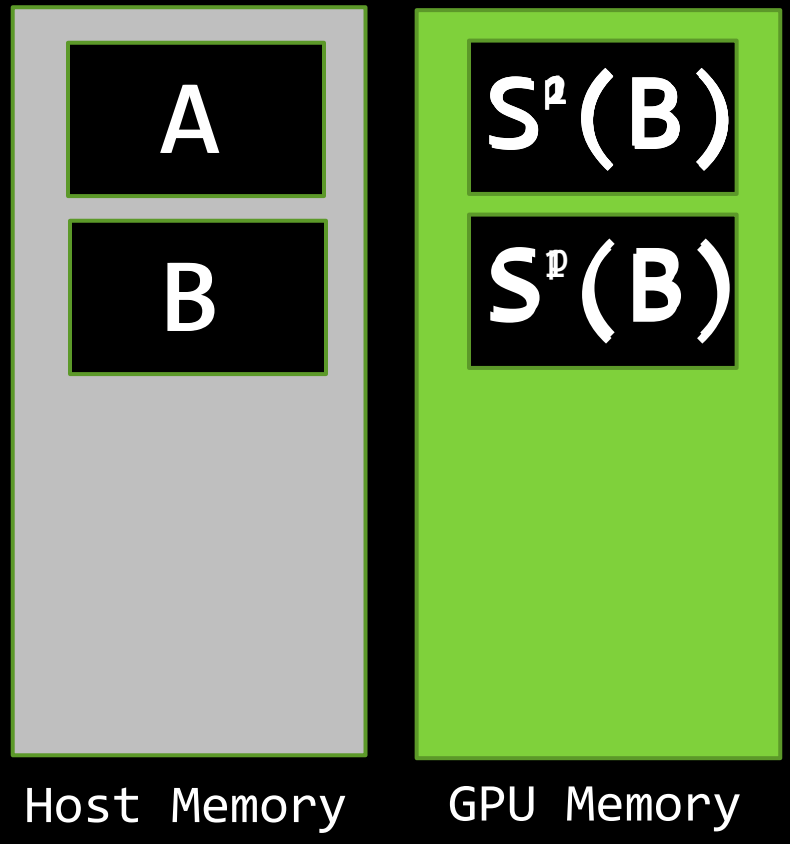
PGI Accelerator compute region

```
for (iter = 1; iter <= niters; ++iter){
  #pragma acc region
  {
  for (i = 1; i < n-1; ++i){
    for (j = 1; j < m-1; ++j){
      a[i][j]=w0*b[i][j]+
              w1*(b[i-1][j]+b[i+1][j]+
                  b[i][j-1]+b[i][j+1])+
              w2*(b[i-1][j-1]+b[i-1][j+1]+
                  b[i+1][j-1]+b[i+1][j+1]);
  } }
  for( i = 1; i < n-1; ++i )
    for( j = 1; j < m-1; ++j )
      b[i][j] = a[i][j];
  }
}
```



PGI Accelerator data region

```
#pragma acc data region \
        copy(b[0:n-1][0:m-1]) \
        local(a[0:n-1][0:m-1])
{
for (iter = 1; iter <= p; ++iter){
  #pragma acc region
  {
  for (i = 1; i < n-1; ++i){
    for (j = 1; j < m-1; ++j){
      a[i][j]=w0*b[i][j]+
              w1*(b[i-1][j]+b[i+1][j]+
                  b[i][j-1]+b[i][j+1])+
              w2*(b[i-1][j-1]+b[i-1][j+1]+
                  b[i+1][j-1]+b[i+1][j+1]);
  } }
  for( i = 1; i < n-1; ++i )
    for( j = 1; j < m-1; ++j )
      b[i][j] = a[i][j];
  }
}
}
```



OpenACC™ API

- Open Standard of Accelerator Directives based on PGI Accelerator Model and OpenMP
- Open to all vendors
- Founding members PGI, NVIDIA, Cray, and CAPS
- http://www.openacc-standard.org

PGI Accelerator Model

```
#pragma acc data region copy(b[0:n*m-1]) local(a[0:n*m-1])
{
  for (iter = 1; iter <= p; ++iter){
    #pragma acc region
    {
      for (i = 1; i < n-1; ++i)
        for (j = 1; j < m-1; ++j){
          a[i*m+j]=w0*b[i*m+j]+
                   w1*(b[(i-1)*m+j]+b[(i+1)*m+j]+
                       b[i*m+j-1]+b[i*m+j+1])+
                   w2*(b[(i-1)*m+j-1]+b[(i-1)*m+j+1]+
                       b[(i+1)*m+j-1]+b[(i+1)*m+j+1]);
        }
      for( i = 1; i < n-1; ++i )
        for( j = 1; j < m-1; ++j )
          b[i*m+j] = a[i*m+j];
    }
  }
}
```

OpenACC™ API

```
#pragma acc data copy(b[0:n*m]) create(a[0:n*m])
{
  for (iter = 1; iter <= p; ++iter){
    #pragma acc kernels
    {
      for (i = 1; i < n-1; ++i)
        for (j = 1; j < m-1; ++j){
          a[i*m+j]=w0*b[i*m+j]+
                   w1*(b[(i-1)*m+j]+b[(i+1)*m+j]+
                       b[i*m+j-1]+b[i*m+j+1])+
                   w2*(b[(i-1)*m+j-1]+b[(i-1)*m+j+1]+
                       b[(i+1)*m+j-1]+b[(i+1)*m+j+1]);
        }
      for( i = 1; i < n-1; ++i )
        for( j = 1; j < m-1; ++j )
          b[i*m+j] = a[i*m+j];
    }  }  }
```

OpenACC™ API

```
#pragma acc data copy(b[0:n*m]) create(a[0:n*m])
{
  for (iter = 1; iter <= p; ++iter){
    #pragma acc parallel
    {
      #pragma acc loop collapse(2)
      for (i = 1; i < n-1; ++i)
        for (j = 1; j < m-1; ++j){
          a[i*m+j]=w0*b[i*m+j]+
                   w1*(b[(i-1)*m+j]+b[(i+1)*m+j]+
                       b[i*m+j-1]+b[i*m+j+1])+
                   w2*(b[(i-1)*m+j-1]+b[(i-1)*m+j+1]+
                       b[(i+1)*m+j-1]+b[(i+1)*m+j+1]);
        }
      #pragma acc loop collapse(2)
      for( i = 1; i < n-1; ++i )
        for( j = 1; j < m-1; ++j )
          b[i*m+j] = a[i*m+j];
    }  }  }
```

OpenACC™ API

```
#pragma acc data present(b) create(a[0:n*m])
{
  for (iter = 1; iter <= p; ++iter){
    #pragma acc parallel
    {
      #pragma acc loop collapse(2)
      for (i = 1; i < n-1; ++i)
        for (j = 1; j < m-1; ++j){
          a[i*m+j]=w0*b[i*m+j]+
                   w1*(b[(i-1)*m+j]+b[(i+1)*m+j]+
                       b[i*m+j-1]+b[i*m+j+1])+
                   w2*(b[(i-1)*m+j-1]+b[(i-1)*m+j+1]+
                       b[(i+1)*m+j-1]+b[(i+1)*m+j+1]);
        }
      #pragma acc loop collapse(2)
      for( i = 1; i < n-1; ++i )
        for( j = 1; j < m-1; ++j )
          b[i*m+j] = a[i*m+j];
    }  }  }
```

Let’s look at a more complex example …

SEISMIC_CPML

SEISMIC_CPML is a set of ten open-source Fortran90 programs to solve the two-dimensional or three-dimensional isotropic or anisotropic elastic, viscoelastic or poroelastic wave equation using a finite-difference method with Convolutional or Auxiliary Perfectly Matched Layer (C-PML or ADE-PML) conditions, developed by Dimitri Komatitsch and Roland Martin from University of Pau, France.*

Accelerated source used is taken from the 3D elastic finite-difference code in velocity and stress formulation with Convolutional-PML (C-PML) absorbing conditions.

* http://www.geodynamics.org/cig/software/seismic_cpml

Step 1: Evaluation

- Is my algorithm right for a GPU?
  - SEISMIC_CPML models seismic waves through the earth. Has an outer time step loop with 9 inner parallel loops. Uses MPI and OpenMP parallelization
- Good candidate for the GPU, but not ideal.

Step 2: Add Compute Regions

```
!$acc region
  do k = kmin,kmax
    do j = NPOINTS_PML+1, NY-NPOINTS_PML
      do i = NPOINTS_PML+1, NX-NPOINTS_PML
      total_energy_kinetic = total_energy_kinetic + 0.5d0 * rho*(vx(i,j,k)**2 + vy(i,j,k)**2 + vz(i,j,k)**2)
      epsilon_xx = ((lambda + 2.d0*mu) * sigmaxx(i,j,k) - lambda * sigmayy(i,j,k) - lambda*sigmazz(i,j,k)) / (4.d0 * mu * (lambda + mu))
      epsilon_yy = ((lambda + 2.d0*mu) * sigmayy(i,j,k) - lambda * sigmaxx(i,j,k) - lambda*sigmazz(i,j,k)) / (4.d0 * mu * (lambda + mu))
      epsilon_zz = ((lambda + 2.d0*mu) * sigmazz(i,j,k) - lambda * sigmaxx(i,j,k) - lambda*sigmayy(i,j,k)) / (4.d0 * mu * (lambda + mu))
      epsilon_xy = sigmaxy(i,j,k) / (2.d0 * mu)
      epsilon_xz = sigmaxz(i,j,k) / (2.d0 * mu)
      epsilon_yz = sigmayz(i,j,k) / (2.d0 * mu)
      total_energy_potential = total_energy_potential + 0.5d0 * (epsilon_xx * sigmaxx(i,j,k) + epsilon_yy * sigmayy(i,j,k) + &
        epsilon_yy * sigmayy(i,j,k)+ 2.d0 * epsilon_xy * sigmaxy(i,j,k) + 2.d0*epsilon_xz * sigmaxz(i,j,k)+2.d0*epsilon_yz * sigmayz(i,j,k))
      enddo
    enddo
  enddo
!$acc end region
```

Compiler Feedback

```
% pgfortran -Mmpi=mpich2 -fast -ta=nvidia -Minfo=accel
seismic_CPML_3D_isotropic_MPI_ACC_1.F90 -o gpu1.out
seismic_cpml_3d_iso_mpi_openmp:
   1107, Generating copyin(vz(11:91,11:631,kmin:kmax))
         Generating copyin(vy(11:91,11:631,kmin:kmax))
         Generating copyin(vx(11:91,11:631,kmin:kmax))
         Generating copyin(sigmaxx(11:91,11:631,kmin:kmax))
         Generating copyin(sigmayy(11:91,11:631,kmin:kmax))
         Generating copyin(sigmazz(11:91,11:631,kmin:kmax))
         Generating copyin(sigmaxy(11:91,11:631,kmin:kmax))
         Generating copyin(sigmaxz(11:91,11:631,kmin:kmax))
         Generating copyin(sigmayz(11:91,11:631,kmin:kmax))
         Generating compute capability 1.3 binary
         Generating compute capability 2.0 binary
```

```
1108, Loop is parallelizable
1109, Loop is parallelizable
1110, Loop is parallelizable
      Accelerator kernel generated
    1108, !$acc do parallel, vector(4)
    1109, !$acc do parallel, vector(4)
    1110, !$acc do vector(16)
          Using register for 'sigmayz'
          Using register for 'sigmaxz'
          Using register for 'sigmaxy'
          Using register for 'sigmazz'
          Using register for 'sigmayy'
          Using register for 'sigmaxx'
          CC 2.0 : 43 registers; 2056 shared, 140 constant,
                   0 local memory bytes; 33% occupancy
    1116, Sum reduction generated for total_energy_kinetic
    1134, Sum reduction generated for total_energy_potential
```

Initial Timings

Version	MPI Processes	OpenMP Threads	GPUs	Time (sec)	Approx. Programming Time (min)
Original MPI/OMP	2	4	0	948	
ACC Step 1	2	0	2	3599	10

System Info:

4 Core Intel Core-i7 920 Running at 2.67Ghz

Includes 2 Tesla C2070 GPUs

Problem Size: 101x641x128

Why the slowdown?

Step 3: Optimize Data Movement

```
!$acc data region                                                    &
!$acc      copyin(a_x_half,b_x_half,k_x_half,                        &
!$acc                a_y_half,b_y_half,k_y_half,                     &
!$acc                a_z_half,b_z_half,k_z_half,                     &
!$acc                a_x,a_y,a_z,b_x,b_y,b_z,k_x,k_y,k_z,            &
!$acc                sigmaxx,sigmaxz,sigmaxy,sigmayy,sigmayz,sigmazz, &
!$acc                memory_dvx_dx,memory_dvy_dx,memory_dvz_dx,      &
!$acc                memory_dvx_dy,memory_dvy_dy,memory_dvz_dy,      &
!$acc                memory_dvx_dz,memory_dvy_dz,memory_dvz_dz,      &
!$acc                memory_dsigmaxx_dx, memory_dsigmaxy_dy,         &
!$acc                memory_dsigmaxz_dz, memory_dsigmaxy_dx,         &
!$acc                memory_dsigmaxz_dx, memory_dsigmayz_dy,         &
!$acc                memory_dsigmayy_dy, memory_dsigmayz_dz,         &
!$acc                memory_dsigmazz_dz)

  do it = 1,NSTEP
.... Cut ....
  enddo
!$acc end data region
```

Timings Continued

Version	MPI Processes	OpenMP Threads	GPUs	Time (sec)	Approx. Programming Time (min)
Original MPI/OMP	2	4	0	948	
ACC Step 1	2	0	2	3599	10
ACC Step 2	2	0	2	194	180

System Info:

4 Core Intel Core-i7 920 Running at 2.67Ghz

Includes 2 Tesla C2070 GPUs

Data movement time only 4 seconds!

Step 4: Fine Tune Schedule

```
!$acc do vector(4)
  do k=k2begin,NZ_LOCAL
   kglobal = k + offset_k
!$acc do parallel, vector(4)
   do j=2,NY
!$acc do parallel, vector(16)
     do i=1,NX-1
      value_dvx_dx = (vx(i+1,j,k)-vx(i,j,k)) * ONE_OVER_DELTAX
      value_dvy_dy = (vy(i,j,k)-vy(i,j-1,k)) * ONE_OVER_DELTAY
      value_dvz_dz = (vz(i,j,k)-vz(i,j,k-1)) * ONE_OVER_DELTAZ
```

Final Timings

Version	MPI Processes	OpenMP Threads	GPUs	Time (sec)	Approx. Programming Time (min)
Original MPI/OMP	2	4	0	948	
ACC Step 1	2	0	2	3599	10
ACC Step 2	2	0	2	194	180
ACC Step 3	2	0	2	167	120

5x in 5 Hours!

Cluster Timings

Version	Size	MPI Processes	OpenMP Threads	GPUs	Time (sec)
MPI/OMP	101x641x512	16	256	0	607
MPI/ACC	101x641x512	16	0	16	186
MPI/OMP	101x641x1024	16	256	0	1920
MPI/ACC	101x641x1024	16	0	16	335

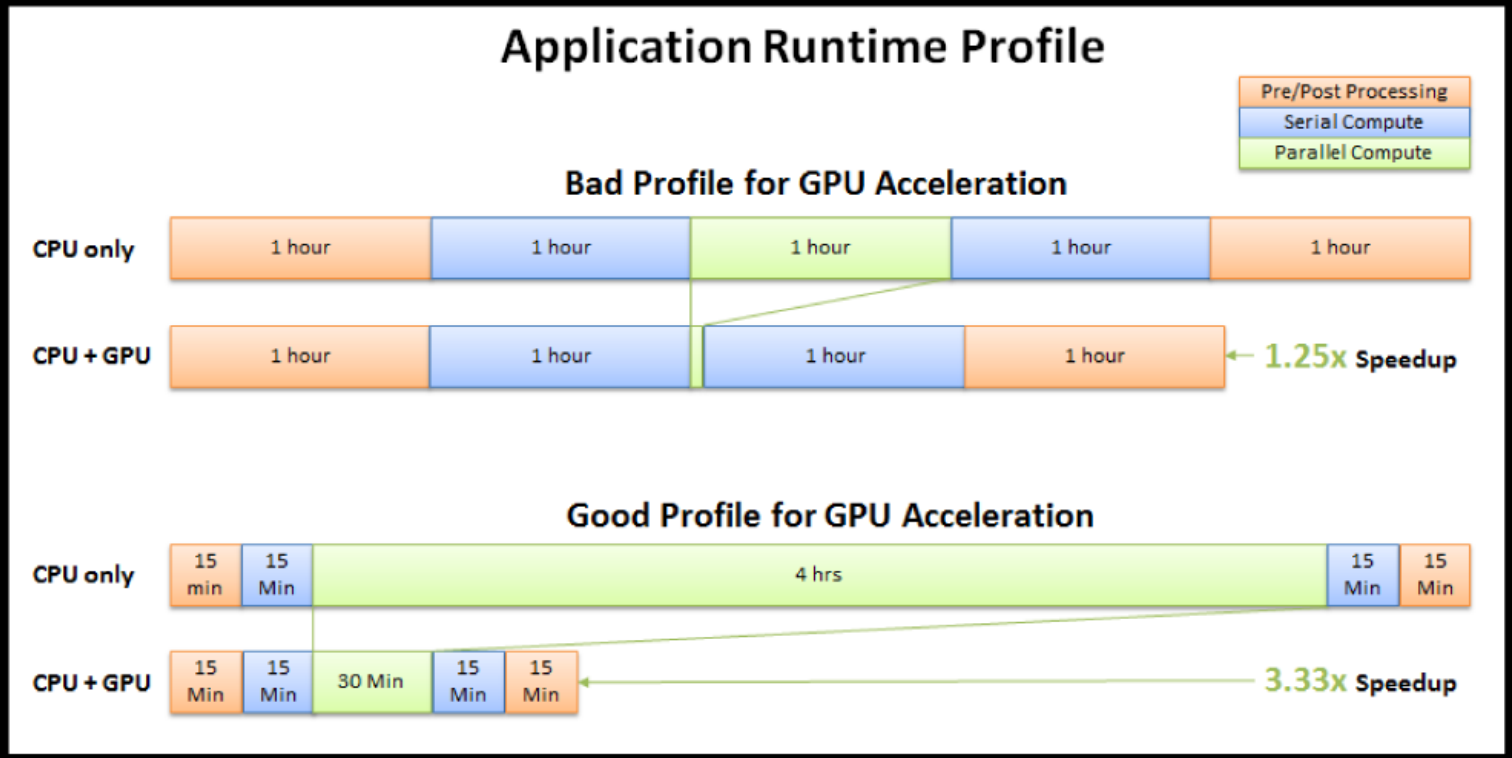
System Info: 16 Nodes

Four socket AMD 8356 @2.3GHZ (16 cores per node)

Tesla C1060

Still 5x!

Are PGI Accelerator directives right for your application?



Are PGI Accelerator directives right for your application?

- Loops with large aggregate iteration counts
- Some loops must be fully parallel
- Loops must be rectangular
- Locality to enable use of GPU shared memory

2x in 4 Weeks Program Overview

Benefit

- Double your app performance in less than a month
- Overall improved parallel code, even on CPU only runs
- If you don't achieve 2x, we will provide consultation to help at no charge to accelerate your code*

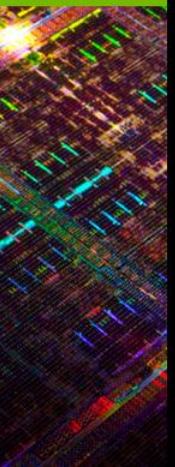
Offer

- 30 day free trial license of PGI Accelerator Compiler
- First 100 participants who respond to weekly surveys and write a summary at the end of 4 weeks will receive a free perpetual PGI Accelerator compiler license compliments of NVIDIA

*Must meet eligibility requirements to qualify for four hours of consultation with an expert in PGI Accelerator or CUDA.

Key Value of the PGI Compilers

PGI compilers are designed to enable performance-portable programming for all Manycore and GPU Accelerator-based systems *without having to re-program for each new successive hardware advancement.*



Register for the Next GTC Express

Debugging CUDA with TotalView

Chris Gottbrath, Principal Product Manager, TotalView, Rogue Wave Software

Wednesday, February 22, 2012, 9:00 AM PST

By attending the webinar you'll learn how to

- Install and run the TotalView Debugger with CUDA across several programming examples
- Debug both the CPU and the GPU code in applications that use CUDA
- Display how your logical threads are being mapped to hardware
- Navigate kernel threads using either hardware or logical coordinates

Register at www.gputechconf.com