ADVANCED OpenACC COURSE

Lecture 1: Advanced Profiling, May 19, 2016

NVIDIA

Course Objective:

Enable *you* to scale *your* applications on multiple GPUs and optimize with profiler tools

Course Syllabus

May 19: Advanced Profiling of OpenACC Code

May 26: Office Hours

June 2: Advanced multi-GPU Programming with MPI and OpenACC

June 9: Office Hours

Recordings:
https://developer.nvidia.com/openacc-advanced-course

ADVANCED PROFILING OF OpenACC CODE

Lecture 1: Ty McKercher, NVIDIA

NVIDIA

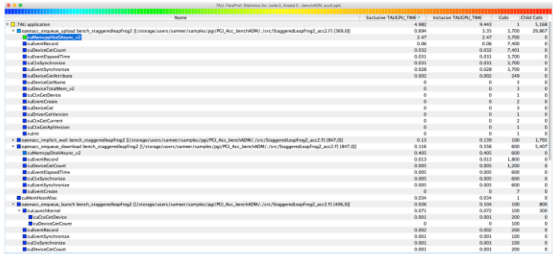
3 WAYS TO ACCELERATE APPLICATIONS

Applications

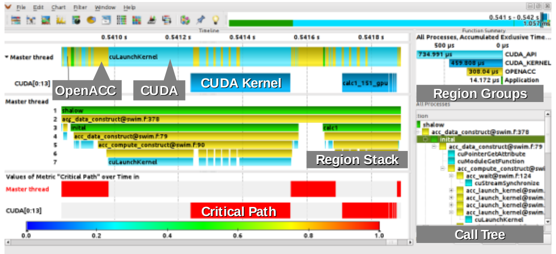
Libraries	OpenACC Directives	Programming Languages
"Drop-in" Acceleration	Easily Accelerate Applications	Maximum Flexibility

OpenACC Profiling Tools

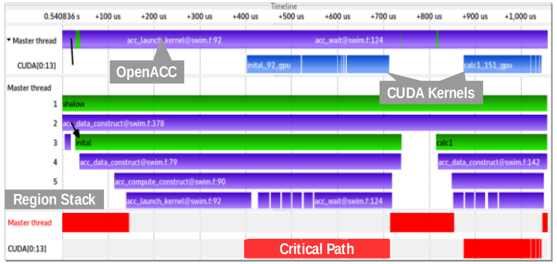
OpenACC Tools Interface



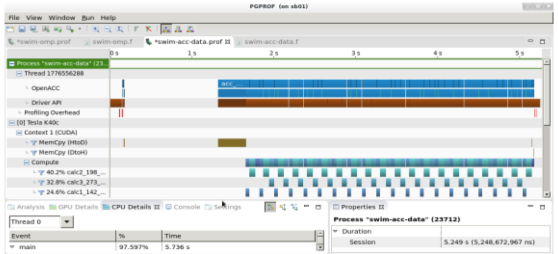
TAU



Vampir



Score-P



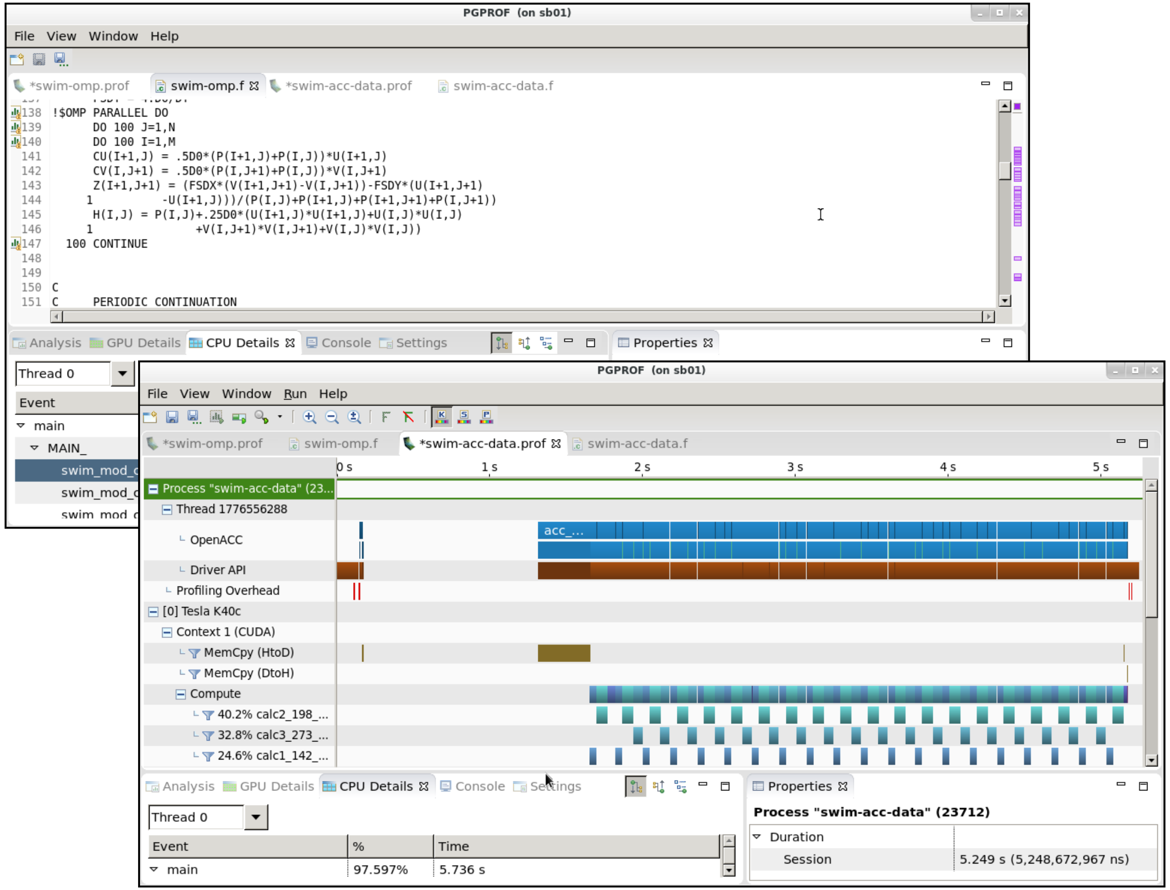
PGPROF

“The OpenACC profiling interface lets us easily measure and investigate implicit operations introduced by the compiler or runtime. Implementation was straightforward and basically worked out of the box.”

Dipl.-Ing. Robert Dietrich, T-U Dresden

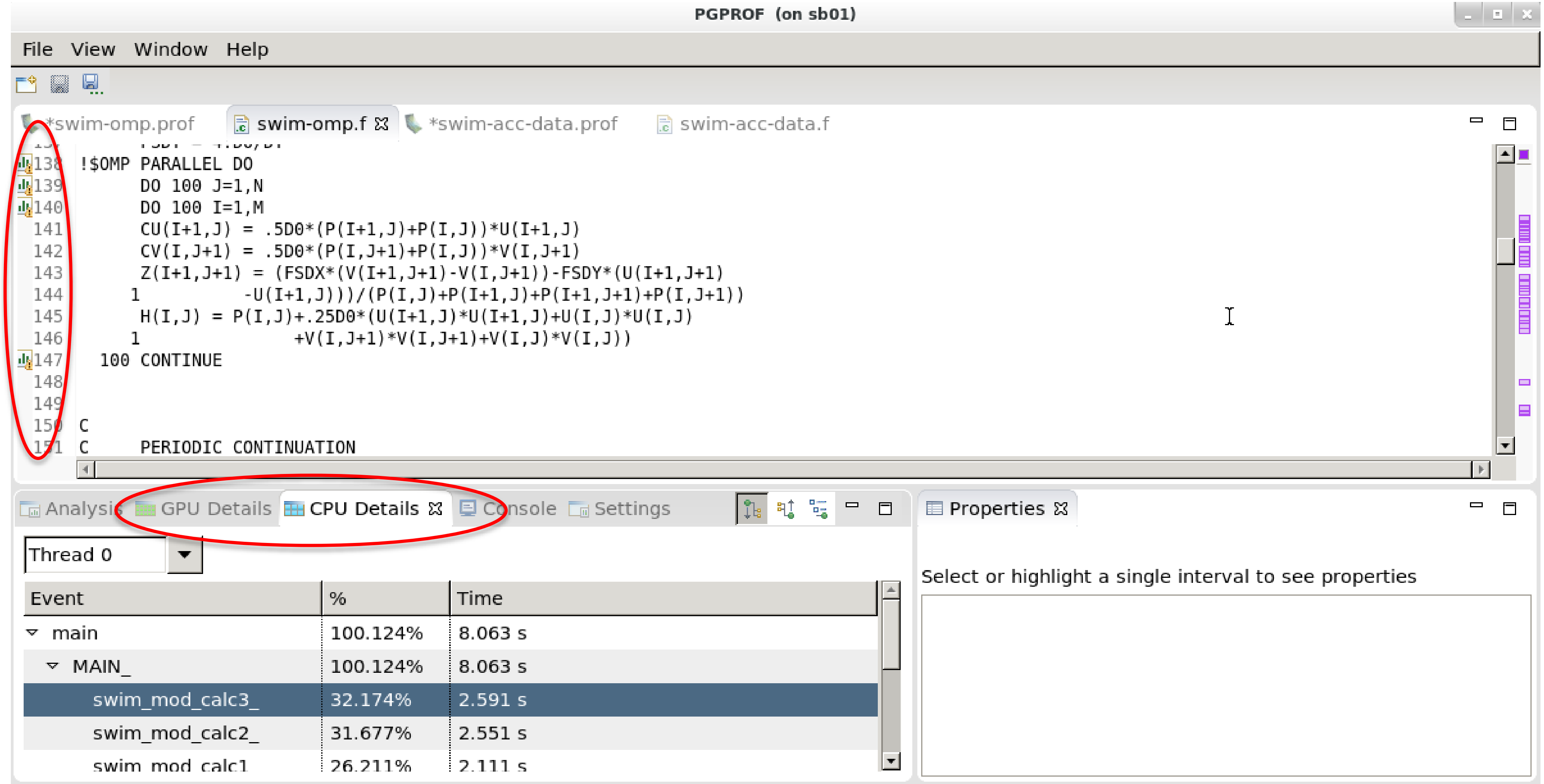
PGPROF: OpenACC CPU AND GPU PROFILER

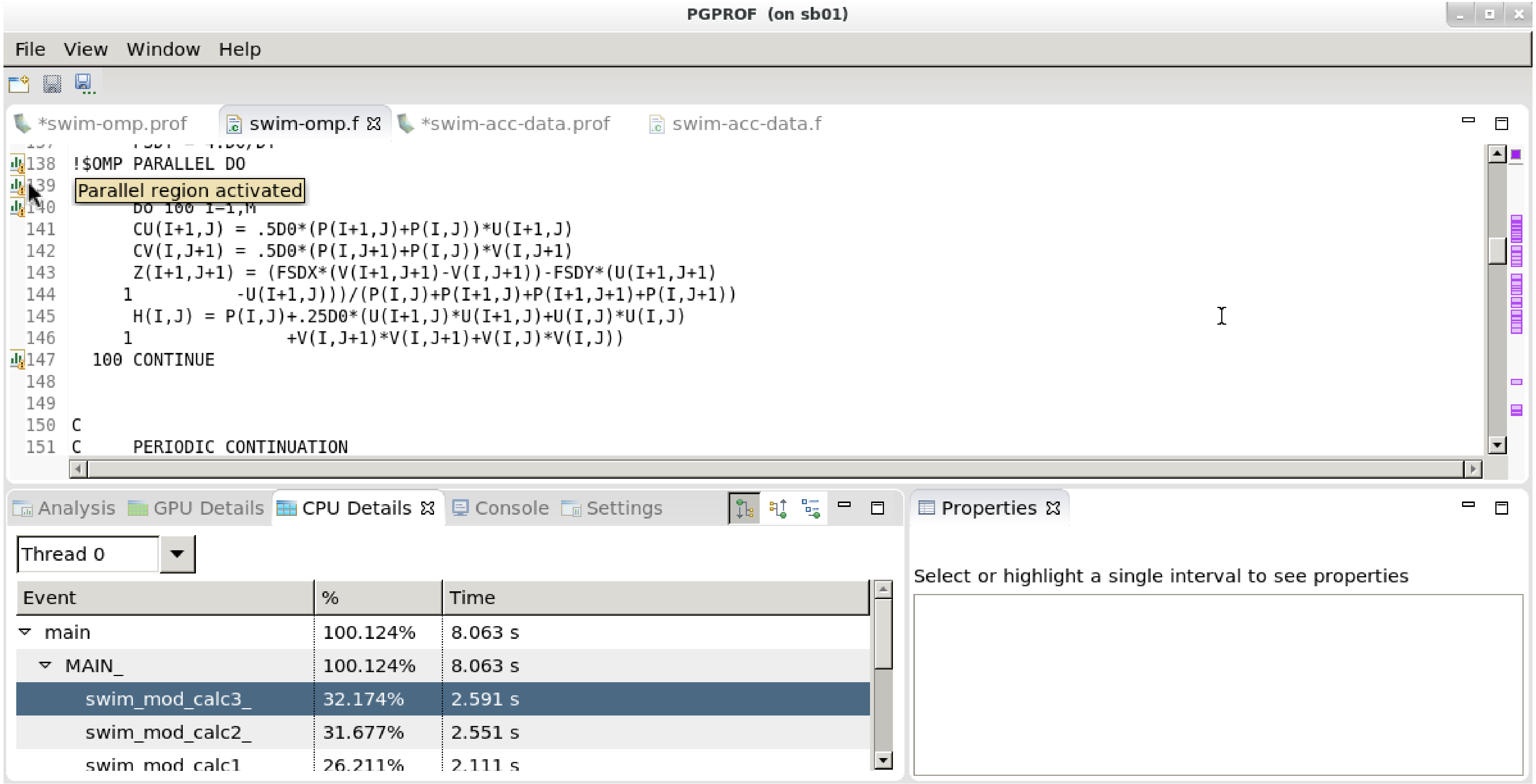
Available with the latest OpenACC Toolkit

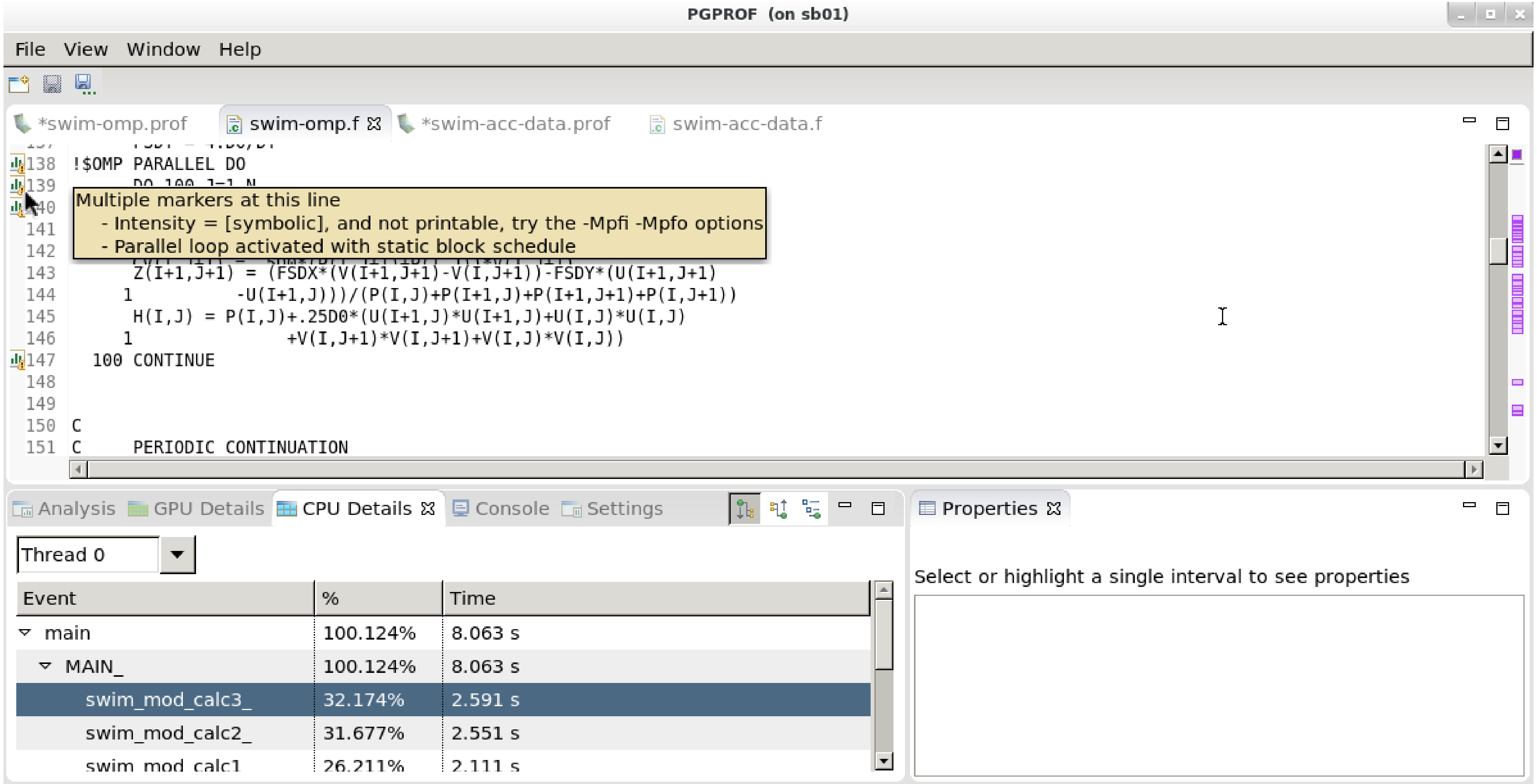


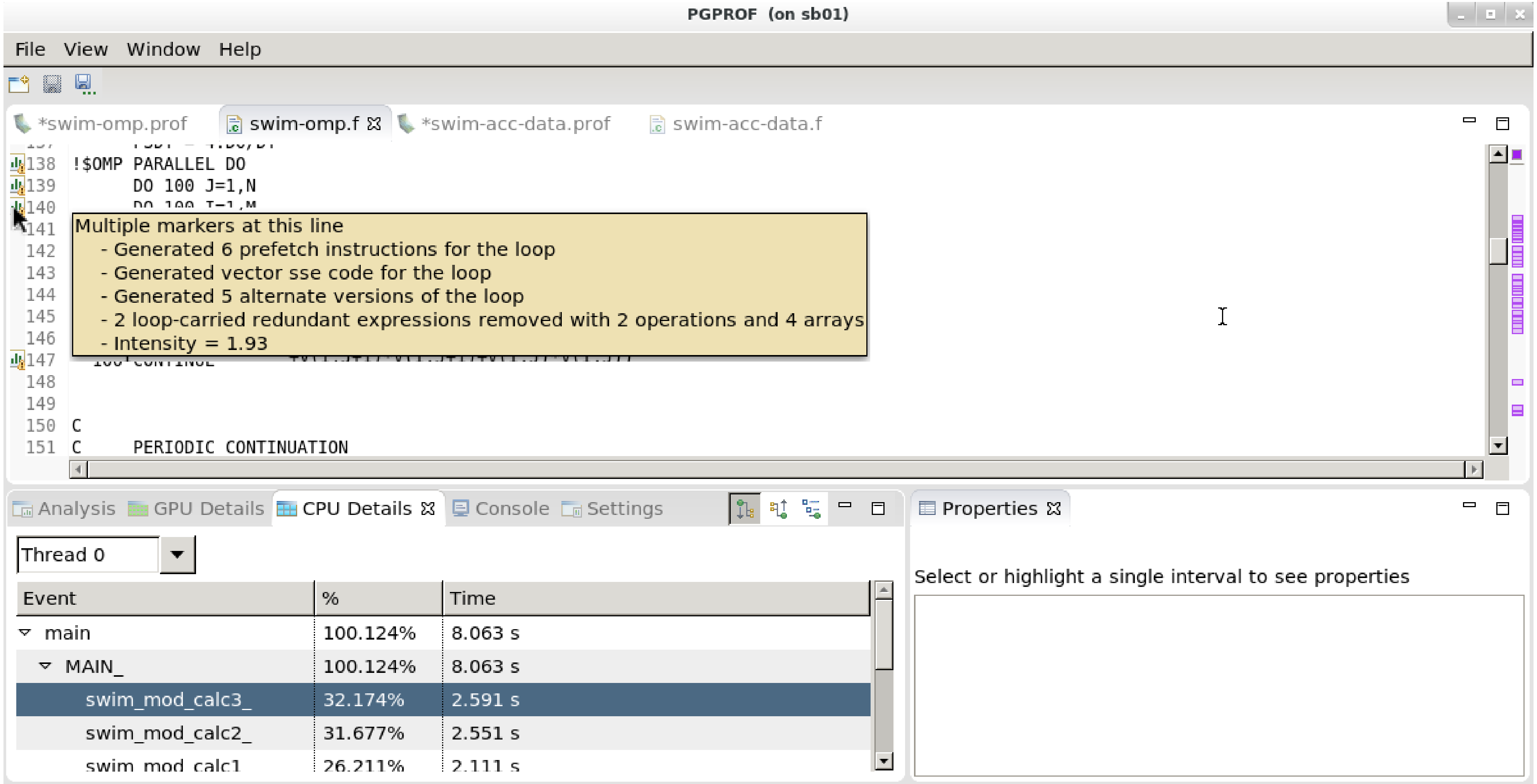
- For 64-bit multicore processor-based systems with or without accelerators
- Supports thread-level OpenMP profiling
- Supports profiling OpenACC and CUDA Fortran codes on NVIDIA CUDA-enabled GPU accelerators
- Graphical and command-line user interfaces
- Function level (routine) and source code line level profiling
- Comprehensive built-in help facilities

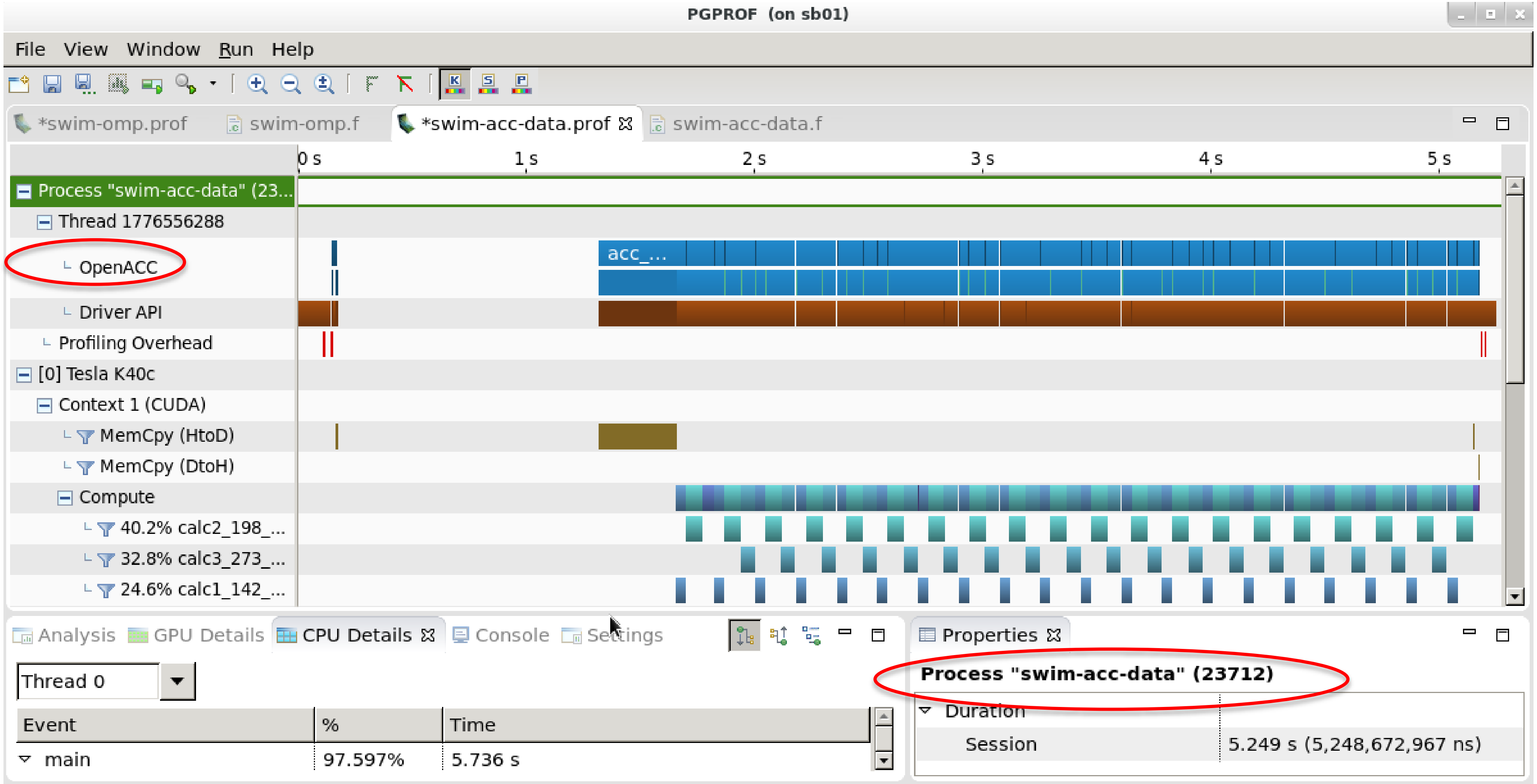
http://www.pgroup.com/products/pgprof.htm


PGPROF (on sb01)
File View Window Help
*swim-omp.prof
swim-omp.f
*swim-acc-data.prof
swim-acc-data.f
138 !$OMP PARALLEL DO
139 DO 100 J=1,N
140 DO 100 I=1,M
141 CU(I+1,J) = .5D0*(P(I+1,J)+P(I,J))*U(I+1,J)
142 CV(I,J+1) = .5D0*(P(I,J+1)+P(I,J))*V(I,J+1)
143 Z(I+1,J+1) = (FSDX*(V(I+1,J+1)-V(I,J+1))-FSDY*(U(I+1,J+1)
144 1 -U(I+1,J)))/(P(I,J)+P(I+1,J)+P(I+1,J+1)+P(I,J+1))
145 H(I,J) = P(I,J)+.25D0*(U(I+1,J)*U(I+1,J)+U(I,J)*U(I,J)
146 1 +V(I,J+1)*V(I,J+1)+V(I,J)*V(I,J))
147 100 CONTINUE
148
149
150 C
151 C PERIODIC CONTINUATION
Analysis
GPU Details
CPU Details
Console
Settings
Thread 0
Event
%
Time
main
100.124%
8.063 s
MAIN_
100.124%
8.063 s
swim_mod_calc3_
32.174%
2.591 s
swim_mod_calc2_
31.677%
2.551 s
swim_mod_calc1
26.211%
2.111 s
Properties
Select or highlight a single interval to see properties


PGPROF (on sb01)
File View Window Help
*swim-omp.prof
swim-omp.f
*swim-acc-data.prof
swim-acc-data.f
138 !$OMP PARALLEL DO
Parallel region activated
141 CU(I+1,J) = .5D0*(P(I+1,J)+P(I,J))*U(I+1,J)
142 CV(I,J+1) = .5D0*(P(I,J+1)+P(I,J))*V(I,J+1)
143 Z(I+1,J+1) = (FSDX*(V(I+1,J+1)-V(I,J+1))-FSDY*(U(I+1,J+1)
144 1 -U(I+1,J)))/(P(I,J)+P(I+1,J)+P(I+1,J+1)+P(I,J+1))
145 H(I,J) = P(I,J)+.25D0*(U(I+1,J)*U(I+1,J)+U(I,J)*U(I,J)
146 1 +V(I,J+1)*V(I,J+1)+V(I,J)*V(I,J))
147 100 CONTINUE
150 C
151 C PERIODIC CONTINUATION
Analysis
GPU Details
CPU Details
Console
Settings
Properties
Thread 0
Event
%
Time
main 100.124% 8.063 s
MAIN_ 100.124% 8.063 s
swim_mod_calc3_ 32.174% 2.591 s
swim_mod_calc2_ 31.677% 2.551 s
Select or highlight a single interval to see properties


PGPROF (on sb01)
File View Window Help
*swim-omp.prof
swim-omp.f
*swim-acc-data.prof
swim-acc-data.f
138 !$OMP PARALLEL DO
Multiple markers at this line
- Intensity = [symbolic], and not printable, try the -Mpfi -Mpfo options
- Parallel loop activated with static block schedule
143 Z(I+1,J+1) = (FSDX*(V(I+1,J+1)-V(I,J+1))-FSDY*(U(I+1,J+1)
144 1 -U(I+1,J)))/(P(I,J)+P(I+1,J)+P(I+1,J+1)+P(I,J+1))
145 H(I,J) = P(I,J)+.25D0*(U(I+1,J)*U(I+1,J)+U(I,J)*U(I,J)
146 1 +V(I,J+1)*V(I,J+1)+V(I,J)*V(I,J))
147 100 CONTINUE
150 C
151 C PERIODIC CONTINUATION
Analysis
GPU Details
CPU Details
Console
Settings
Properties
Thread 0
Select or highlight a single interval to see properties
Event
%
Time
main 100.124% 8.063 s
MAIN_ 100.124% 8.063 s
swim_mod_calc3_ 32.174% 2.591 s
swim_mod_calc2_ 31.677% 2.551 s
swim_mod_calc1 26.211% 2.111 s


PGPROF (on sb01)
File View Window Help
*swim-omp.prof
swim-omp.f
*swim-acc-data.prof
swim-acc-data.f
138 !$OMP PARALLEL DO
139 DO 100 J=1,N
Multiple markers at this line
- Generated 6 prefetch instructions for the loop
- Generated vector sse code for the loop
- Generated 5 alternate versions of the loop
- 2 loop-carried redundant expressions removed with 2 operations and 4 arrays
- Intensity = 1.93
150 C
151 C PERIODIC CONTINUATION
Analysis
GPU Details
CPU Details
Console
Settings
Properties
Thread 0
Event
%
Time
main
100.124%
8.063 s
MAIN_
100.124%
8.063 s
swim_mod_calc3_
32.174%
2.591 s
swim_mod_calc2_
31.677%
2.551 s
Select or highlight a single interval to see properties


PGPROF (on sb01)
File View Window Run Help
*swim-omp.prof
swim-omp.f
*swim-acc-data.prof
swim-acc-data.f
0 s
1 s
2 s
3 s
4 s
5 s
Process "swim-acc-data" (23...
Thread 1776556288
OpenACC
acc_...
Driver API
Profiling Overhead
[0] Tesla K40c
Context 1 (CUDA)
MemCpy (HtoD)
MemCpy (DtoH)
Compute
40.2% calc2_198_...
32.8% calc3_273_...
24.6% calc1_142_...
Analysis
GPU Details
CPU Details
Console
Settings
Thread 0
Event
%
Time
main
97.597%
5.736 s
Properties
Process "swim-acc-data" (23712)
Duration
Session
5.249 s (5,248,672,967 ns)

More on Visual Profiler

Office Hour on May 26^{th}

PGPROF Quick Start Guide: http://www.pgroup.com/resources/pgprof-quickstart.htm

PGPROF User's Guide*: http://www.pgroup.com/doc/pgprofug.pdf

* Updated version by May 25^{th}

Agenda

Profiling application - Getting Started

Acceleration with Managed Memory

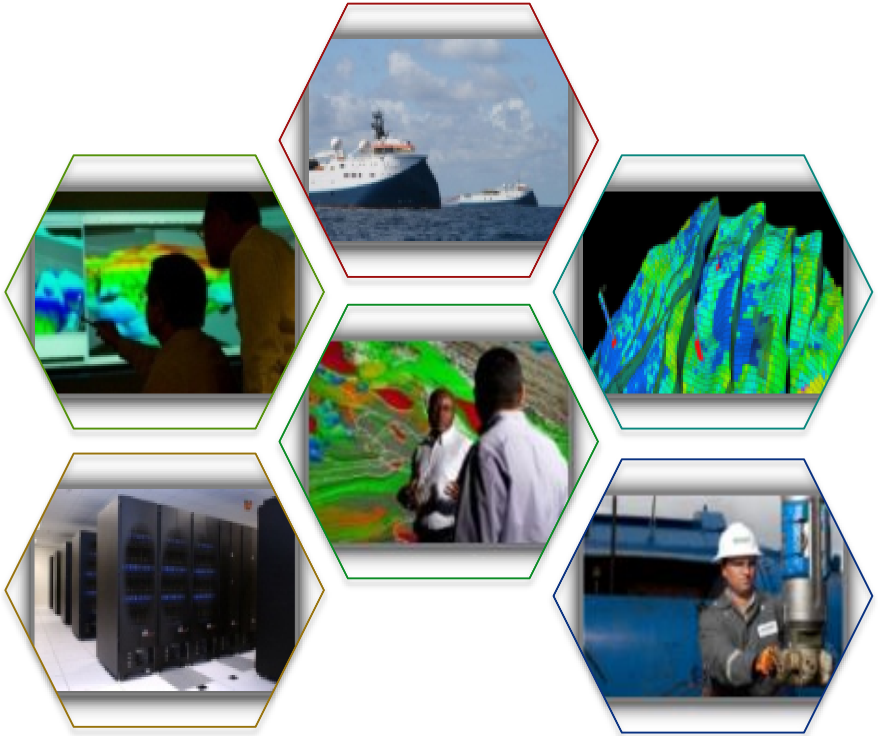
Optimization with Data Directives

Multicore comparison

Seismic Unix Configuration

Profiling Application – Getting Started

EXPLORATION & PRODUCTION WORKFLOW



1. Acquire Seismic Data
2. Process Seismic Data
3. Interpret Seismic Data
4. Characterize Reservoirs
5. Simulate Reservoirs
6. Drill Wells

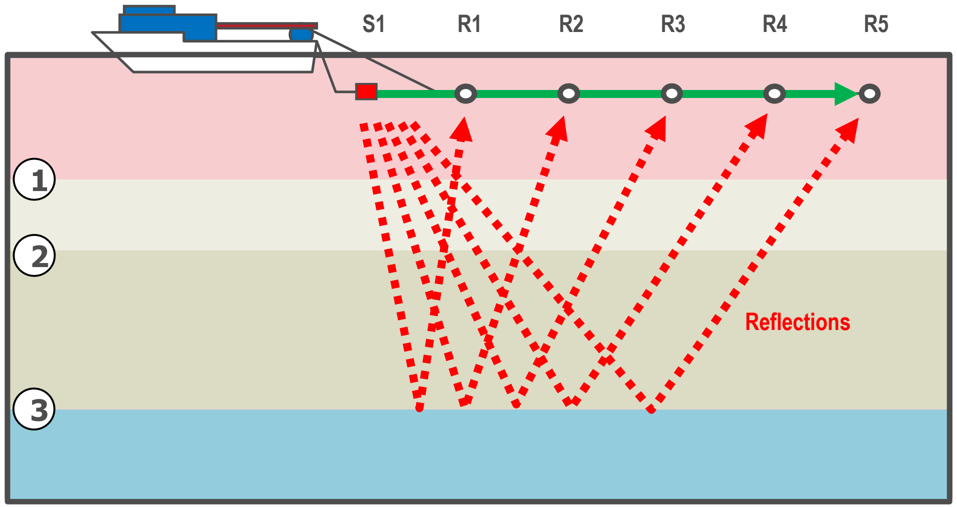
Images courtesy Schlumberger

EXPLORATION & PRODUCTION WORKFLOW

- Acquire Seismic Data
- Process Seismic Data
- Interpret Seismic Data
- Characterize Reservoirs
- Simulate Reservoirs
- Drill Wells

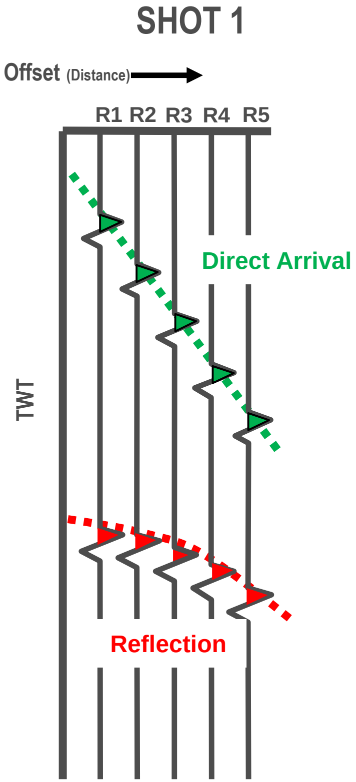
Images courtesy Schlumberger

ACQUIRE SEISMIC DATA



For each shot, reflections are recorded in 5 receivers

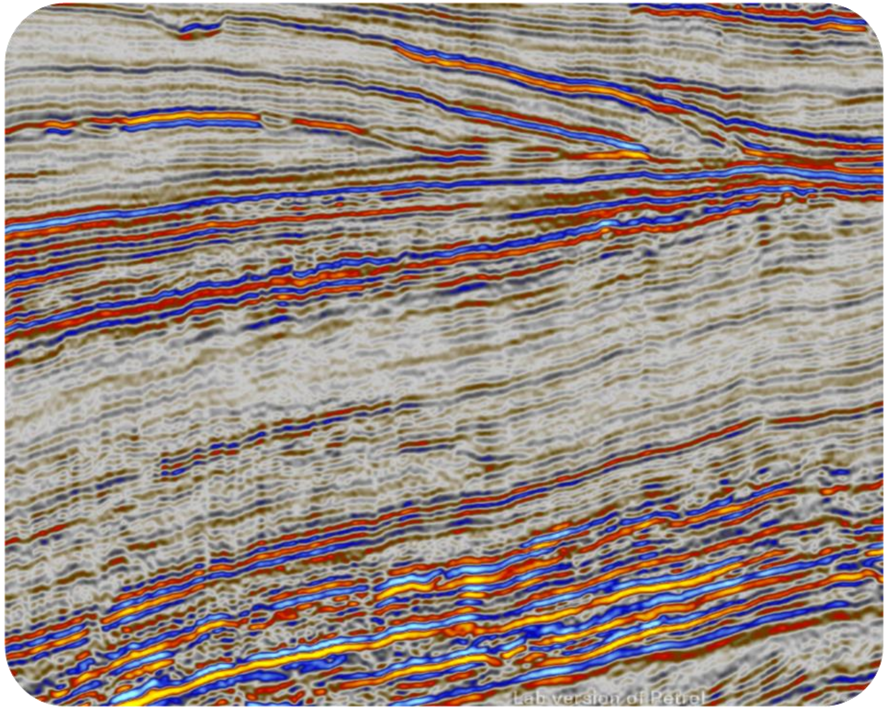
There are 5 'bounce' points along interface 3



Images courtesy: Schlumberger, GTC 2015

REAL ROCK vs SEISMIC REFLECTION

Imaging, signal processing, filtering, ray tracing



Images courtesy: Schlumberger, GTC 2015

USE KNOWN MODEL

Verify accuracy of imaging algorithms

- Industry standard dataset
- Compare against known truth
- Refine seismic algorithms
- Improve confidence in drilling decisions

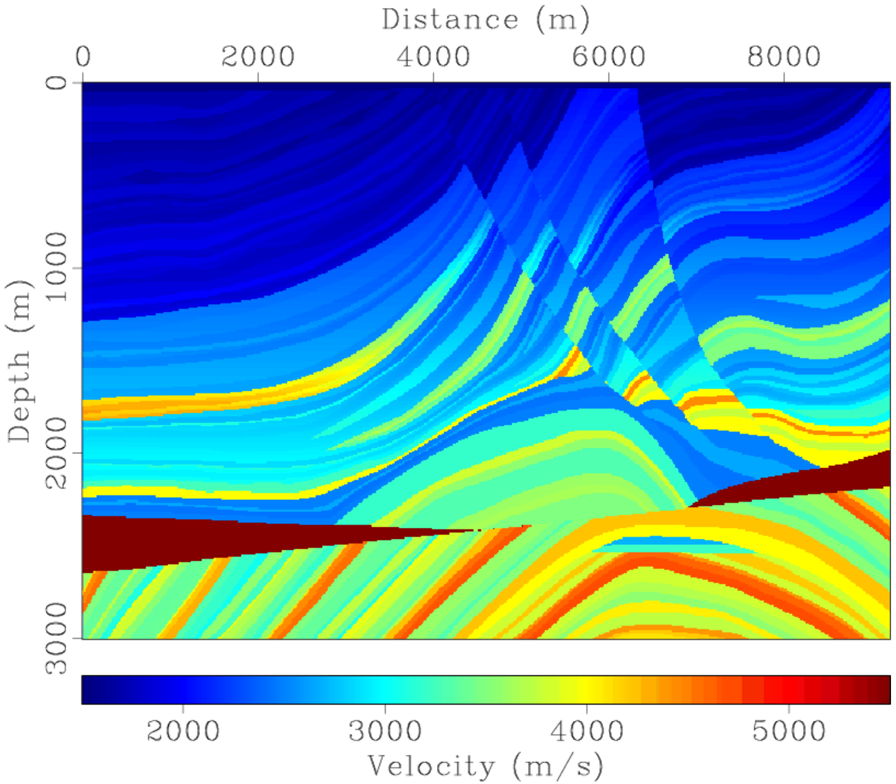
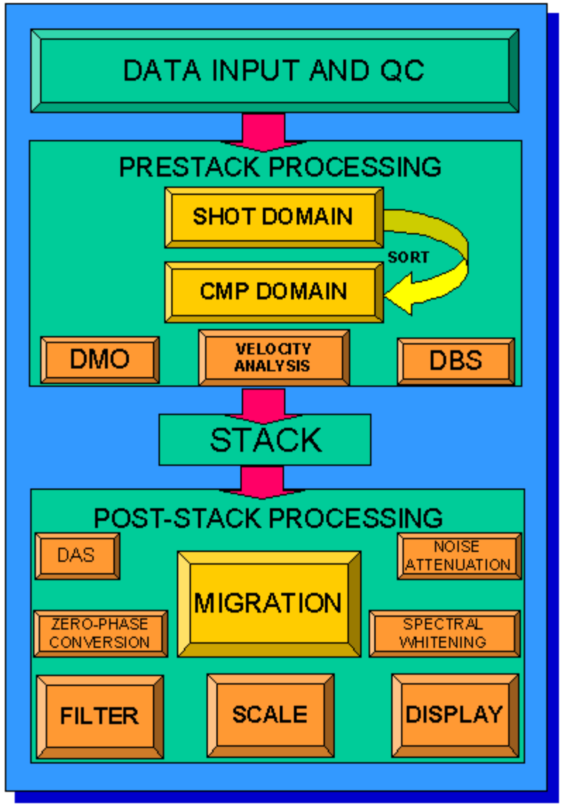


Image sources: The Marmousi velocity model

PROCESS SEISMIC DATA

Open Source Seismic*Unix Package

Typical Seismic Processing Flow



- Iterative process
  - Can take several months
- Migration most time consuming
  - Collapse diffractions
  - Correctly position dip events
- Trusted tool: Kirchhoff Migration

Image sources: Seismic Processing The Marmousi velocity model

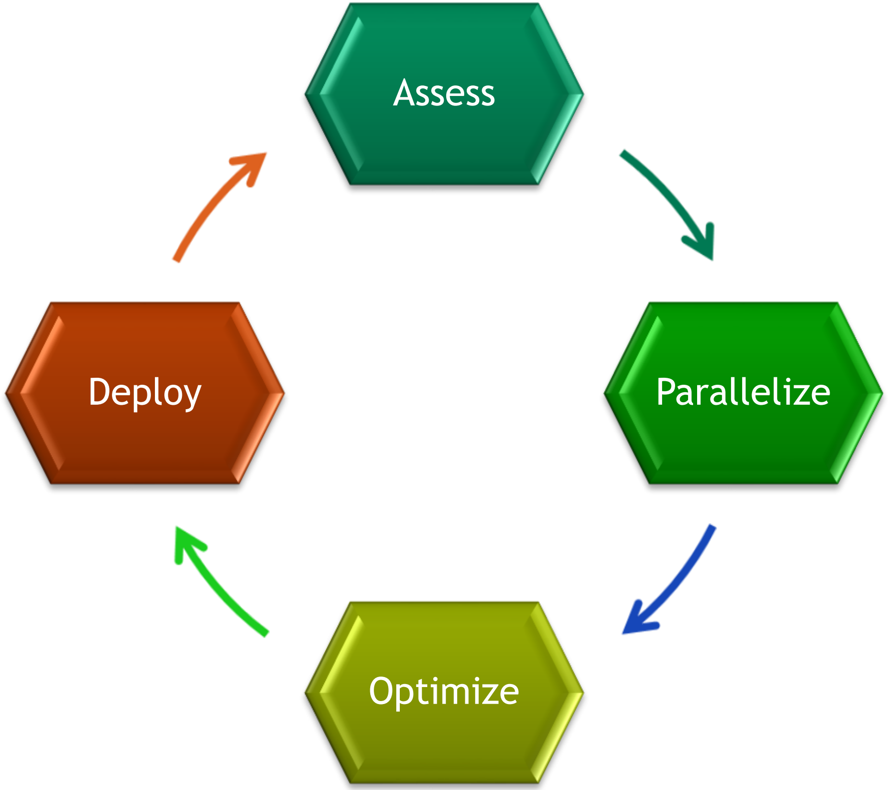
PSEUDO-CODE FOR KIRCHHOFF MIGRATION

sukdmig2d

- Update residual travel times
- Loop over traces
    - Combine travel times (sum2)
    - Migrate trace (mig2d)
        - Low-pass filter trace
        - Compute amplitudes
        - Interpolate along lateral
        - Interpolate along depth
- Write output

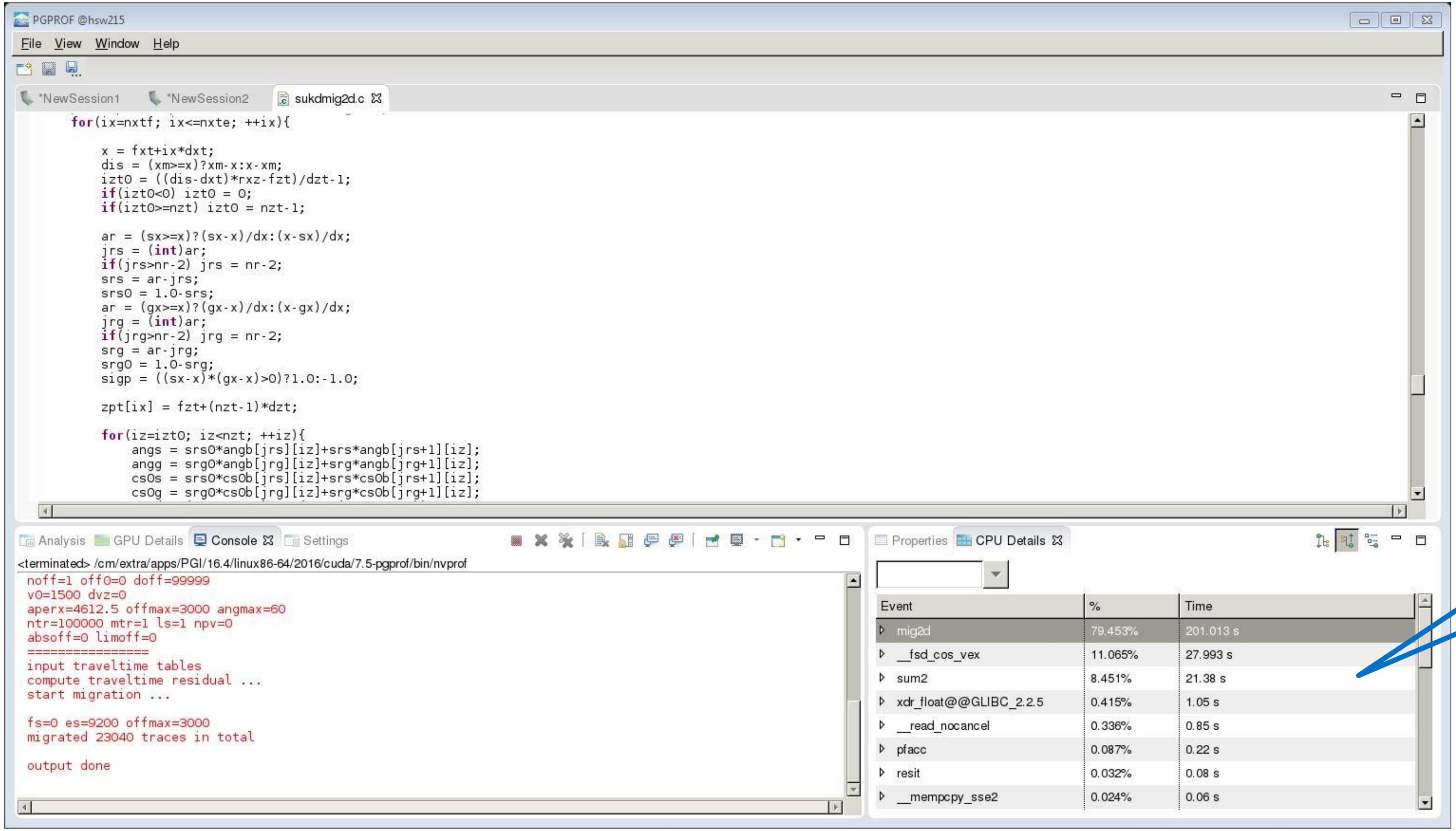
```
main() {
...
  resit();

  do {
    ...
    sum2(…,ttab, tt);
    sum2(…,tsum, tsum);

    mig2d(…, mig, …, mig1, …);

  } while (fget(tr) && jtr<ntr);

  fput(tr);

}
```

HOW YOU ACCELERATE CODE WITH GPUs



ASSESS BASELINE CPU PROFILE

Use `pgprof sukdmig2d`



Function	Percent Runtime
mig2d	79%
sum2	8.5 %
resit	< 1%

RESULTS

SUKDMIG2D	Configuration	Model Size	Cores	Elapsed Time (s)	Speed up
CPU Only (Baseline)	2x E5-2698 v3 2.30GHz	2301 x 751	1	218	1.00

Acceleration with Managed Memory

PARALLELIZE

Parallel Directives

#pragma

Parallelize for loops

Vectorize

Compiler vectorizes inner loops

```
mig2d:
  #pragma acc parallel for
  for (ix=nxtf; ix<=nxte; ++ix) {
 . . .
     #pragma acc loop
     for (iz=izt0; iz<nzt; ++iz) {
 . . .
```

PARALLELIZE

Parallel Directives

restrict on pointers!
limits aliasing
www.wikipedia.org/wiki/Restrict

#pragma
Parallelize outer for loops

Compiler parallelizes inner loop

```
void sum2(int nx, int nz,float a1,float a2,
  float ** restrict t1, float ** restrict t2, float **
restrict t)
{
   int ix,iz;

   #pragma acc parallel for
   for(ix=0; ix < nx; ++ix)
   {
       for(iz=0; iz < nz; ++iz)
           t[ix][iz] = a1*t1[ix][iz]+a2*t2[ix][iz];
   }
}
```

PARALLELIZE

Resolve Errors!

Parallel Directives

- #pragma
- Parallelize for outer loop

Parallelize inner loops

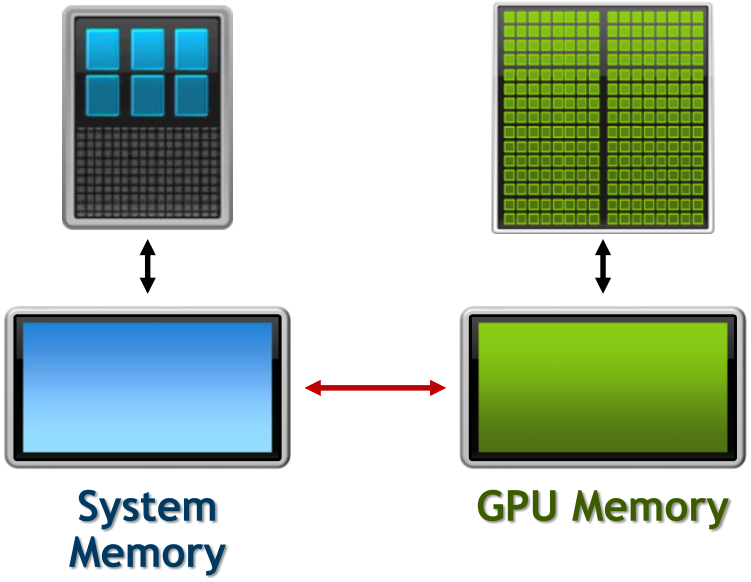
- Resolve loop carried depend
- Add acc loop directive

```
Resit (managed):
537, Accelerator kernel generated
     Generating Tesla code
     538, #pragma acc loop gang /* blockIdx.x */
     553, #pragma acc loop vector(128) /* threadIdx.x */
540, Loop carried dependence of t->-> prevents parallelization
     Loop carried backward dependence of t->-> prevents vector
```

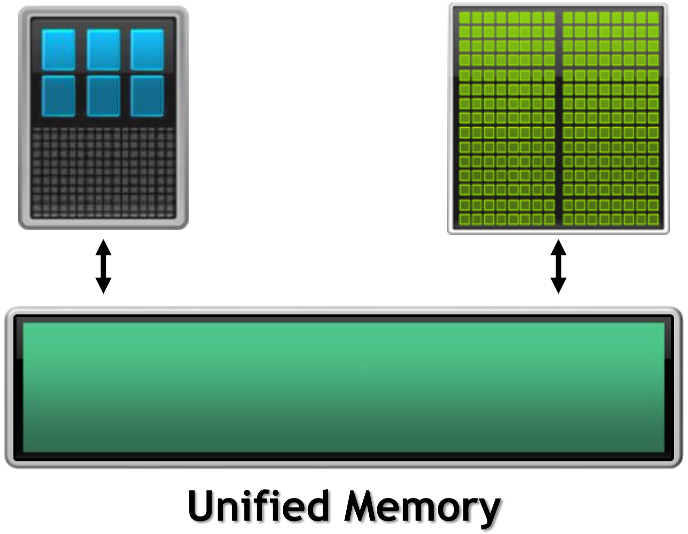
```
Resit:
  #pragma acc parallel for
  for (ix=0; ix<nx; ++ix)
  {
    #pragma acc loop
    for (is=0; is<ns; ++is)
    {
. . .
          #pragma acc loop
          for (iz=0; iz<nz; ++iz)
              t[ix][iz] -= sr0*tb[jr][iz]+sr*tb[jr+1][iz];
```

UNIFIED MEMORY IMPROVES PRODUCTIVITY

Previous Developer View



Developer View With Unified Memory



OPENACC AND UNIFIED MEMORY

All heap allocations are in managed memory (Unified Memory Heap)

Pointers can be used on GPU and CPU

Enabled with compiler switch –ta=tesla:**managed**,…

More Info at „OpenACC and CUDA Unified Memory", by Michael Wolfe, PGI Compiler Engineer: https://www.pgroup.com/lit/articles/insider/v6n2a4.htm

OpenACC and Unified Memory

Advantages

No need for any data clauses

No need to fully understand application data flow and allocation logic

Incremental profiler-driven acceleration possible

Outlook to GPU programming on Pascal

RESULTS

SUKDMIG2D	Configuration	Model Size	Cores	Elapsed Time (s)	Speed up
CPU Only (Baseline)	2x E5-2698 v3 2.30GHz	2301 x 751	1	218	1.00
OpenACC (Managed)	1x K40 GPU	2301 x 751	2880	46	**4.70**

RESULTS

SUKDMIG2D	Configuration	Model Size	Cores	Elapsed Time (s)	Speed up
CPU Only (Baseline)	2x E5-2698 v3 2.30GHz	2301 x 751	1	218	1.00
OpenACC (Managed)	1x K40 GPU	2301 x 751	2880	46	**4.70**

Now optimize using the Verbose output from compiler!

Optimization with Data Directives

OPTIMIZATION

Compile

```
pgcc -acc \
  -ta=tesla:managed
```

Profile !

```
pgprof <managed binary>
```

```
==55246== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 42.82%  4.03645s     23040  175.19us  121.12us  196.38us  mig2d_787_gpu
 28.79%  2.71389s     23040  117.79us  80.800us  135.68us  mig2d_726_gpu
 27.35%  2.57762s     69120  37.291us  33.248us  42.240us  sum2_571_gpu
  1.00%  93.936ms     23040  4.0770us  3.2000us  12.992us  [CUDA memcpy HtoD]
  0.04%  3.4627ms         1  3.4627ms  3.4627ms  3.4627ms  resit_537_gpu
  0.00%  126.14us         1  126.14us  126.14us  126.14us  timeb_592_gpu

==55246== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 30.16%  11.5982s    230423  50.334us     118ns  3.9101ms  cuMemFree
 29.21%  11.2327s    230429  48.746us  10.132us  12.821ms  cuMemAllocManaged
 27.15%  10.4430s    253444  41.204us  1.0420us  3.4680ms  cuStreamSynchronize
 10.42%  4.00751s    115202  34.786us  5.4290us  99.805ms  cuLaunchKernel
  1.13%  433.50ms   1428513     303ns     141ns  429.42us  cuPointerGetAttrib
  0.81%  310.55ms         1  310.55ms  310.55ms  310.55ms  cuDevicePrimary…
  0.71%  273.10ms     23040  11.853us  7.3210us  409.13us  cuMemcpyHtoDAsync
  0.33%  125.36ms         1  125.36ms  125.36ms  125.36ms  cuDevicePrimary…
  0.06%  24.165ms         1  24.165ms  24.165ms  24.165ms  cuMemHostAlloc…
  0.02%  9.5668ms         1  9.5668ms  9.5668ms  9.5668ms  cuMemFreeHost
  0.00%  534.34us         1  534.34us  534.34us  534.34us  cuMemAllocHost
  0.00%  461.71us         1  461.71us  461.71us  461.71us  cuModuleLoad..
  0.00%  363.83us         2  181.91us  180.02us  183.81us  cuMemAlloc
```

OPTIMIZATION

Managed Compile

- Verbose output
- Guided enhancements
- Targeted changes

Common Optimizations

- Data Movement
  - Copy, copyin, copyout
  - Create, delete
  - Update
- Loop Collapse

```
main:
 453, Generating update host(mig[:noff][:nxo][:nzo])
 455, Generating update host(mig1[:noff][:1][:1])
 459, Generating update host(mig1[:noff][:nxo][:nzo])

resit:
 539, Generating copyin(ttab[:ns],tb[:][:nz])

sum2:
 571, Generating copyin(t2[:nx][:nz],t1[:nx][:nz])
      Generating copyout(t[:nx][:nz])

mig2d:
 721, Generating copy(ampt1[nxtf:nxte-nxtf+1][:])
     Generating copyin(cssum[nxtf:nxte-nxtf+1][:],tvsum[nxtf:nxte-nxtf+1][
     Generating copy(tmt[nxtf:nxte-nxtf+1][:],ampti[nxtf:nxte-nxtf+1][:])
     Generating copyin(pb[:][:])
     Generating copy(ampt[nxtf:nxte-nxtf+1][:])
     Generating copyin(cs0b[:][:],angb[:][:])
     Generating copy(zpt[nxtf:nxte-nxtf+1])
 782, Generating copy(mig1[nxf:nxe-nxf+1][:])
     Generating copyin(ampt1[:][:], tb[:][:], tsum[:][:], ampt[:][:], ...
     Generating copy(mig[nxf:nxe-nxf+1][:])
     Generating copyin(zpt[:])
```

OPTIMIZATION

Data Movement

Analyze data flow in application
Explicitly use data directives

Move data directive to *main*

Create only when possible

Copyin move data to GPU

Update to move data to host

```
main:
#pragma acc enter data create(tb,pb,cs0b,ang0)
#pragma acc enter data create(tt,tsum)
#pragma acc enter data copyin(mig, ttab)

#pragma acc enter data create(tvsum,csum )
#pragma acc enter data copyin(cs, tv)
#pragma acc enter data copyin(mig1)

After processing:

#pragma acc update host(mig)
#pragma acc update host(mig1)
```

OPTIMIZATION

Data Movement

Explicitly use present for data already on GPU!

Collapse

Increase the threads nx*nz

Present

Data is already on the GPU

Prevent data movement

```
sum2: (managed)
 571, Generating copyin(t2[:nx][:nz],t1[:nx][:nz])
      Generating copyout(t[:nx][:nz])
```

```
void sum2(int nx, int nz,float a1,float a2,
  float ** restrict t1, float ** restrict t2, float **
restrict t)
{
   int ix,iz;

   #pragma  acc parallel for collapse(2) present(t1,t2,t)
   for(ix=0; ix < nx; ++ix)
   {
      for(iz=0; iz < nz; ++iz)
         t[ix][iz] = a1*t1[ix][iz]+a2*t2[ix][iz];
   }
}
```

OPTIMIZATION

Data Movement

Move large data transfers to main i.e. mig, mig1

Minimize Copyin, Copyout

Maximize Create, Present

Prevents data transfers

Use Copyin, Copyout, Copy only when data changes!

Delete happens when leaving scope

```
mig2d:
 721, Generating copy(ampt1[nxtf:nxte-nxtf+1][:])
      Generating copyin(cssum[nxtf:nxte-nxtf+1][:],tvsum[...
      Generating copy(tmt[nxtf:nxte-nxtf+1][:],ampti[...
      Generating copyin(pb[:][:])
      Generating copy(ampt[nxtf:nxte-nxtf+1][:])
      Generating copyin(cs0b[:][:],angb[:][:])
      Generating copy(zpt[nxtf:nxte-nxtf+1])
 782, Generating copy(mig1[nxf:nxe-nxf+1][:])
      Generating copyin(ampt1[:][:], tb[:][:], tsum[:][:], ...
      Generating copy(mig[nxf:nxe-nxf+1][:])
```

```
void mig2d(float * restrict trace, int nt, float ft,...)
{
...
#pragma acc data
  copyin(trace[0:nz],trf[0:nt+2*mtmax]) \
  present(mig, mig1, tb,tsum,tvsum,cssum,pb,... \
  create(tmt[0:nxt][0:nzt], ampt[0:nxt][0:nzt],...
  {
```

OPTIMIZATION

Data Movement

Use present for data already on GPU!

Collapse

Increase the threads nx*ns

Present

Data is already on the GPU
Prevent data movement

```
Resit: (managed)
 539, Generating copyin(ttab[:ns],tb[:][:nz])
```

```
resit:
 ...
  #pragma acc parallel for collapse(2) present(tb, ttab)
  for (ix=0; ix<nx; ++ix)
  {
    for (is=0; is<ns; ++is)
    {
...
            #pragma acc loop
            for (iz=0; iz<nz; ++iz)
                t[ix][iz] -= sr0*tb[jr][iz]+sr*tb[jr+1][iz];
     }
```

OPTIMIZATION

Data Movement

- mig, mig1 data large
  - Move to main
  - Copyin at start
  - Mark as present
  - Copyout for snapshots
- Minimize Copyin, Copyout
- Use create
  - Prevents copy in/out
- Delete happens when leaving scope

```
void mig2d(float * restrict trace, int nt, float
ft,...)
{
...
#pragma acc data
  copyin(trace[0:nz],trf[0:nt+2*mtmax]) \
  present(mig, mig1, tb,tsum,tvsum,cssum,pb,... \
  create(tmt[0:nxt][0:nzt], ampt[0:nxt][0:nzt],...
  {
...
   #pragma acc parallel for
   for (ix=nxtf; ix <= nxte; ++ix) {
...
      #pragma acc loop
      for (iz=izt0; iz < nzt; ++iz) {
...
```

OPTIMIZATION

Compile

`pgcc -acc -ta=tesla`

Profile

`pgprof <tesla binary>`

`mig2d` and `sum2` about the same.

- ✓ cuAllocManged (11s) removed.
- ✓ cuMemFree (11.5s) reduced to milliseconds.

```
==2242== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 41.54%  3.95071s     23040  171.47us  118.88us  192.61us  mig2d_787_gpu
 27.91%  2.65415s     23040  115.20us  78.241us  133.09us  mig2d_726_gpu
 26.27%  2.49826s     69120  36.143us  32.768us  40.416us  sum2_569_gpu
  2.88%  274.19ms     69132  3.9660us  3.5520us  13.120us  __pgi_uacc_cuda_fill_32_gpu
  1.35%  128.68ms     46088  2.7920us  2.4960us  1.6815ms  [CUDA memcpy HtoD]
  0.04%  3.4187ms         1  3.4187ms  3.4187ms  3.4187ms  resit_535_gpu
  0.00%  226.15us         2  113.07us  2.4640us  223.68us  [CUDA memcpy DtoH]
  0.00%  123.43us         1  123.43us  123.43us  123.43us  timeb_592_gpu

==2242== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 85.89%  9.71880s    138246  70.300us  1.8870us  3.4228ms  cuStreamSynchronize
  7.69%  869.62ms    184334  4.7170us  3.4420us  452.72us  cuLaunchKernel
  2.94%  333.00ms         1  333.00ms  333.00ms  333.00ms  cuDevicePrimaryCtxRetain
  1.75%  197.59ms     46088  4.2870us  2.8370us  426.78us  cuMemcpyHtoDAsync
  1.15%  130.58ms         1  130.58ms  130.58ms  130.58ms  cuDevicePrimaryCtxRelease
  0.25%  28.337ms         1  28.337ms  28.337ms  28.337ms  cuMemHostAlloc
  0.20%  23.059ms     46084     500ns     260ns  11.292us  cuPointerGetAttributes
  0.09%  10.027ms         1  10.027ms  10.027ms  10.027ms  cuMemFreeHost
  0.03%  2.9512ms        31  95.199us  2.9220us  300.63us  cuMemAlloc
  0.01%  806.38us         2  403.19us  188.55us  617.83us  cuModuleLoadData
```

No longer compiling with :managed

RESULTS

SUKDMIG2D	Configuration	Model Size	Cores	Elapsed Time (s)	Speed up
CPU Only (Baseline)	2x E5-2698 v3 2.30GHz	2301 x 751	1	218	1.00
OpenACC (GPU Managed)	1x K40 GPU	2301 x 751	2880	46	4.70
OpenACC (GPU Native)	1x K40 GPU	2301 x 751	2880	**12**	**15.60**

Multicore Comparison

OPTIMIZATION

How about Multi-Core / OMP / pthread?

Done!

Re-Compile

```
pgcc –acc -ta=multicore
```

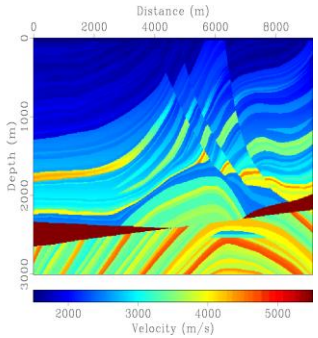
Profile !

```
pgprof
  --cpu-profiling on \
  --cpu-profiling-scope function \
  --cpu-profiling-mode top-down \
  <app> <args>
```

```
======== CPU profiling result (top down):
72.91% main
| 69.84% mig2d
| | 43.19% __pgi_acc_barrier
| | | 43.19% _mp_barrier_tw
| | |   0.02% _mp_pcpu_get_team_lcpu
| | |     0.02% _mp_pcpu_struct
| | |       0.01% __tls_get_addr
| | 0.12% malloc@@GLIBC_2.2.5
| 2.88% sum2
| | 2.79% __pgi_acc_barrier
| | | 2.79% _mp_barrier_tw
| | 0.00% .ACCENTER
| |   0.00% _mp_barrierr
| 0.10% __fsd_cos_vex
| 0.05% __pgi_acc_pexit
| | 0.05% _mp_cpexit
| |   0.05% _mp_barrierw
22.18% _mp_slave
| 22.18% _mp_cslave
|   22.18% _mp_barrier_tw
|     0.02% _mp_pcpu_yield
|       0.02% sched_yield
4.77% __fsd_cos_vex
0.09% filt
-- more --
```

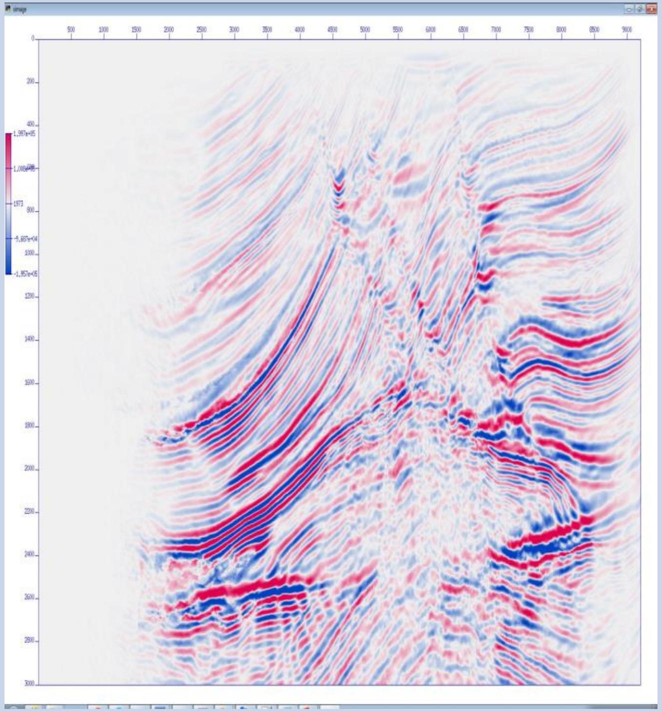
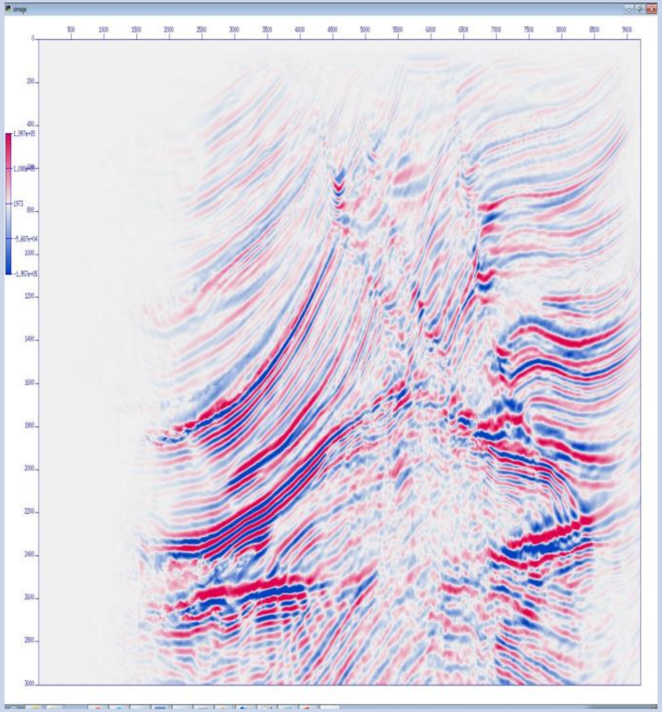
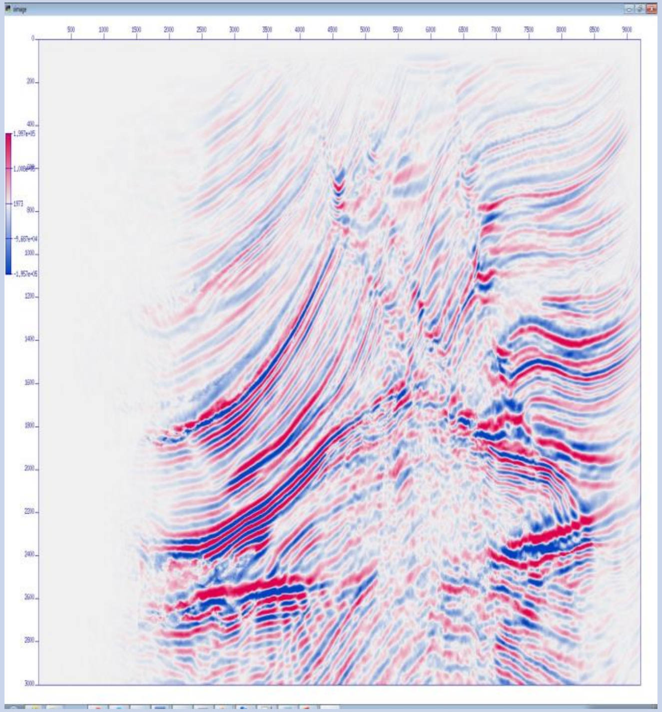
RESULTS

SUKDMIG2D	Configuration	Model Size	Cores	Elapsed Time (s)	Speed up
CPU Only (Baseline)	2x E5-2698 v3 2.30GHz	2301 x 751	1	218	1.00
OpenACC CPU (Multicore)	2x E5-2698 v3 2.30GHz	2301 x 751	16	29	7.50
OpenACC GPU (Managed)	1x K40 GPU	2301 x 751	2880	46	4.70
OpenACC GPU (Native)	1x K40 GPU	2301 x 751	2880	**12**	**15.60**



DEPLOY

How do the results compare?

CPU Only (Baseline)	OpenACC Multicore	OpenACC (GPU)
		

Homework

QWIKLABS: GETTING ACCESS

1. Go to https://developer.nvidia.com/qwiklabs-signup
2. Register with OpenACC promo code to get free access
3. Receive a confirmation email with access instructions

Questions?

Email to openacc@nvidia.com

ACCESS TO HOMEWORK

Qwiklab:

Profile-driven approach to accelerate Seismic application with OpenACC

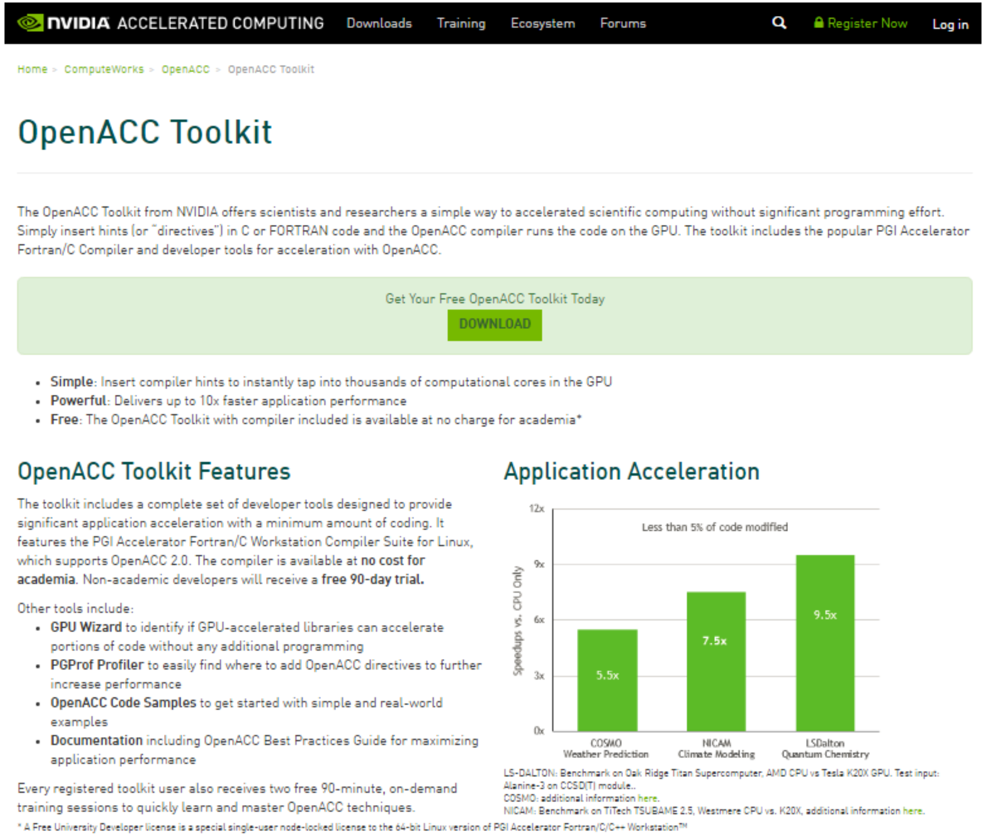
Link: http://bit.ly/oaccnvlab6

Requirements: OpenACC Compiler and CUDA-aware MPI

Link to the source code on github thorough the qwiklab if you want to try it on your machine

INSTALL THE OPENACC TOOLKIT (OPTIONAL)

- Go to developer.nvidia.com/openacc-toolkit
- Register for the OpenACC Toolkit
- Install on your personal machine (Linux Only)
- Free workstation license for academia/90 day free trial for the rest



On Your Own …
Local Seismic Unix Setup

SETUP SEISMIC UNIX

Center for Wave Phenomena

Download Seismic Unix

ftp://ftp.cwp.mines.edu/pub/cwpcodes/cwp_su_all_43R8.tgz

Unpack to ~/cwp

Set environment variables

CWPROOT=~/cwp

PATH=~/cwp/bin:$PATH

Edit Makefile.config, build

Use PGI compilers (CC=pgcc, FC=pgfortran)

OPTC=-g, FFLAGS=$(FOPTS)

SETUP SEISMIC UNIX

Marmousi Datasets

Download Marmousi data, velocity, and density files

http://www.trip.caam.rice.edu/downloads/ieee.tar.gz

Convert SEGY format to SU format

```
#!/bin/bash

segyread tape=data.segy     conv=0 endian=0 > data.su
segyread tape=velocity.segy conv=0 endian=0 > velocity.su

suflip flip=0 < velocity.su > velocity1.su
sustrip < velocity1.su > velocity.h@ ftn=0

suwind < data.su > data1.su tmax=2.9
```

SETUP SEISMIC UNIX

Smooth, build ray trace model, migrate

```
#!/bin/bash

nz = 751
nx = 2301
dz = 4
dx = 4

nt = 750
ntr= 96
dt = 4000
ifile = data1.su
ofile = datamig.su
tfile = tfile
vfile = velocity.h@
```

```
#smoothing
time smooth2 < $vfile n1=$nz n2=$nx r1=20 r2=20
>smoothvel

#raytrace
time rayt2d < smoothvel dt=0.004 nt=751 dz=$dz nz=$nz
dx=$dx nx=$nx fxo=0 dxo=25 nxo=369 fxs=0 dxs=100 nxs=93
>$tfile

#migrate (Example)
sukdmig2d infile=$ifile datain=$ifile outfile=$ofile
dataout=$ofile ttfile=$tfile fzt=0 dzt=4 nzt=751 fxt=0
dxt=25 nxt=369 fs=0 ns=93 ds=100 nzo=751 dzo=4 dxm=25
mtr=1
```

WHERE TO FIND HELP

- OpenACC Course Recordings - https://developer.nvidia.com/openacc-courses
- PGI Website - http://www.pgroup.com/resources
- OpenACC on StackOverflow - http://stackoverflow.com/questions/tagged/openacc
- OpenACC Toolkit - http://developer.nvidia.com/openacc-toolkit
- Parallel Forall Blog - http://devblogs.nvidia.com/parallelforall/
- GPU Technology Conference - http://www.gputechconf.com/
- OpenACC Website - http://openacc.org/

Questions? Email openacc@nvidia.com

Course Syllabus

May 19: Advanced Profiling of OpenACC Code

May 26: Office Hours **<- Visual Profiler**

June 2: Advanced multi-GPU Programming with MPI and OpenACC

June 9: Office Hours

Recordings:
https://developer.nvidia.com/openacc-advanced-course

Questions? Email openacc@nvidia.com