GPU TECHNOLOGY CONFERENCE

A CUDA FORTRAN PORT OF CLOVERLEAF

GREG RUETSCH, NVIDIA

CLOVERLEAF APPLICATION

- Component of Sandia's Mantevo benchmarks
- 2D structured grid hydrodynamic “mini-app”
  - Double precision
  - Explicit compressible Euler equations
  - Finite volume predictor/corrector
  - Bandwidth limited
- CUDA Fortran port based on serial version
  - Single GPU

CUDA FORTRAN PORT

- Goal: ***make minimal changes to source code***
  - Managed memory
    - Single copy of data, implicit data transfers
    - All kernels in time-step loop ported to device
  - CUF kernels (and reduction intrinsics)
    - Implicit kenel generation
  - Implicit textures via LDG instruction
    - No explicit textures or shared memory programming

MANAGED MEMORY

- Memory accessible to both CPU and GPU
- Runtime migrates data between host and device as needed
- Designated by `managed` variable attribute
- Available cc30+, 6.0+ Toolkit, Linux and Windows

MANAGED MEMORY EXAMPLE

```
module kernels
  integer, parameter :: n = 32
contains
  attributes(global) subroutine increment(a)
    integer :: a(*), i
    i = (blockIdx%x-1)*blockDim%x + threadIdx%x
    if (i <= n) a(i) = a(i)+1
  end subroutine increment
end module kernels

program testManaged
  use kernels
  use cudafor
  integer, managed :: a(n)
  integer :: istat
  a = 4
  call increment<<<1,n>>>(a)
  istat = cudaDeviceSynchronize()
  if (all(a==5)) write(*,*) 'OK'
end program testManaged
```

Kernel unchanged

Managed variable attribute

Synchronization required

FLUX_CALC_KERNEL

```
REAL(KIND=8), managed, DIMENSION(x_min-2:x_max+3,y_min-2:y_max+2) :: xarea
…
REAL(KIND=8), managed, DIMENSION(x_min-2:x_max+2,y_min-2:y_max+3) :: vol_flux_y

…

!$cuf kernel do(2) <<<*,*>>>
DO k=y_min,y_max
   DO j=x_min,x_max+1
      vol_flux_x(j,k)=0.25_8*dt*xarea(j,k)                           &
           *(xvel0(j,k)+xvel0(j,k+1)+xvel1(j,k)+xvel1(j,k+1))
   ENDDO
ENDDO
```

MANAGED MEMORY ON MULTI-GPU SYSTEMS

- If peer mappings are not available between any two GPUs, systems falls back to using zero-copy
  - No migration, data resides in host memory
  - PCI transfer for every device access (no caching)
  - Even if single GPU is used

- Environment variables
  - CUDA_VISIBLE_DEVICES
  - CUDA_MANAGED_FORCE_DEVICE_ALLOC

MANAGED MEMORY ON MULTI-GPU SYSTEMS

- Verify peer access using p2pAccess example code included with PGI compilers
  - …/2015/examples/CUDA-Fortran/CUDA-Fortran-Book/chapter4/P2P

- On desktop system with Tesla K20 and Quadro K600
  - 960x960 grid for 87 time steps, on K20

```
…
Wall clock      38.79973196983337
```

  - on K20 with CUDA_VISIBLE_DEVICES=0

```
…
Wall clock      1.249093055725098
```

PORTING CODE USING MANAGED MEMORY

- Declare data used in kernels with **`managed`** attribute
- Insert cudaDeviceSynchronize() after calling device routines (kernels or CUF)
  - Only if managed data are touched from CPU side before another kernel
  - As more code gets ported, these will be removed
- Track kernel execution time, not overall time in initial stages of porting

TIME STEP LOOP ROUTINES

	CUF Kernel	*Explicit Kernel*
accelerate_kernel		✔
advec_cell_kernel		✔
advec_mom_kernel		✔
calc_dt	✔	
calc_dt_kernel		✔
field_summary_kernel	✔	
flux_calc_kernel	✔	
ideal_gas_kernel		✔
PdV		✔
reset_field_kernel	✔	
revert_kernel	✔	
update_halo	✔	
viscosity		✔

CUF KERNELS

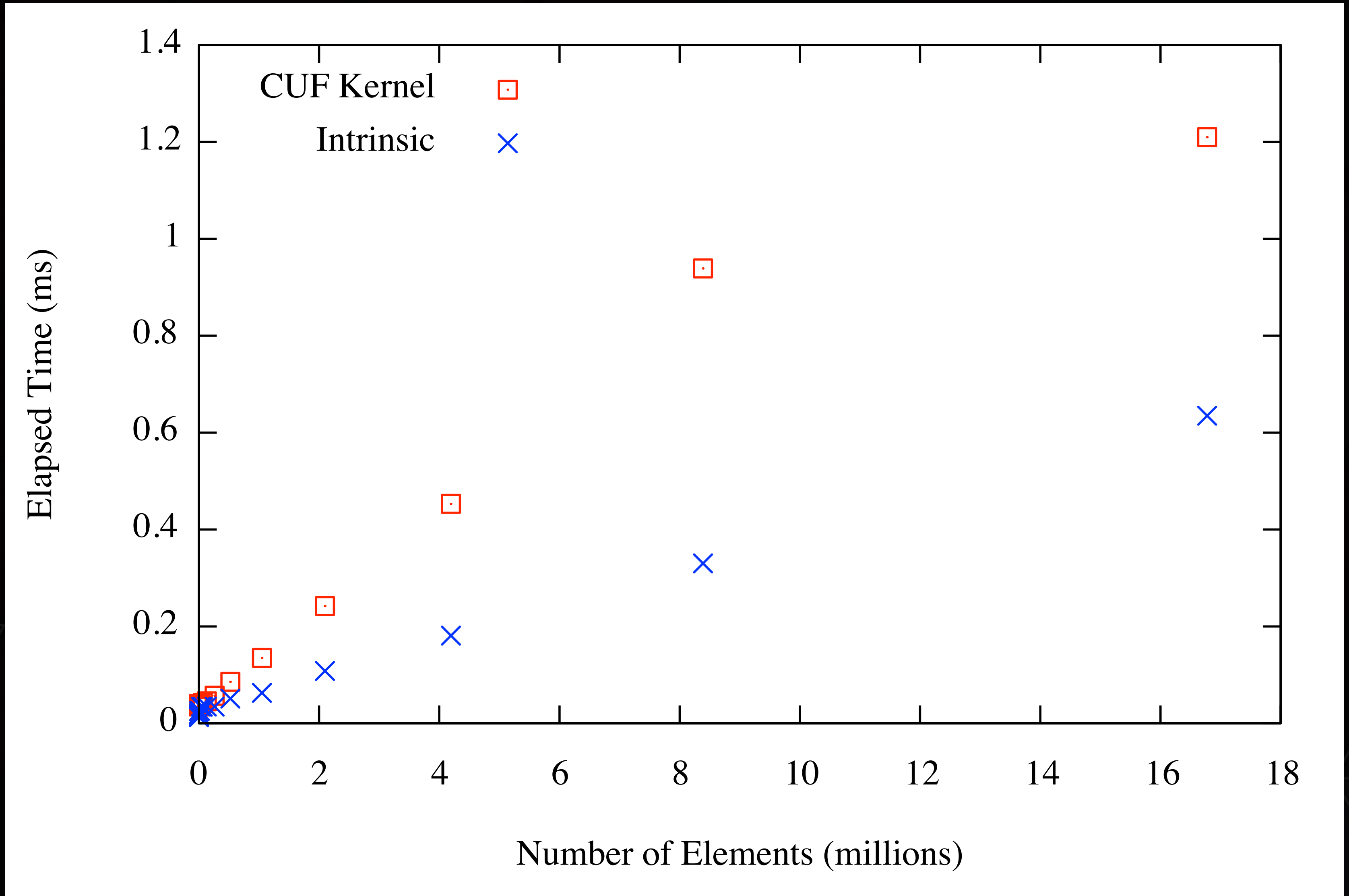
- CUF Kernels
  - Loop directives where compiler generates kernels
  - Used heavily for copies, updates, and reductions in CloverLeaf

```
!$cuf kernel do(2) <<<*,*>>>
DO k=ymin,ymax
   DO j=xmin,xmax
      IF(a(j,k) .LT. dt) dt=a(j,k)
   ENDDO
ENDDO
```

REDUCTION INTRINSICS

- **`maxval`**, **`minval`**, and **`sum`** overloaded to operate on device data from host
  - Requires cc30+ and CUDA 6.0+
- Support for optional arguments **`dim`** and **`mask`** (for **`managed`** data only)
  - generates CUF kernel
- Uses SHFL instruction when no optional arguments and no slice notation

SUM REDUCTION (CUF VS. INTRINSIC)



REDUCTION INTRINSICS

- Control location of reduction intrinsic execution on **managed** data via rename option in "**use cudafor**" statement

```
program reductionRename
  use cudafor, gpusum => sum
  implicit none
  integer, managed :: m(3000)
  integer :: istat
  m = 1
  istat = cudaDeviceSynchronize()
  write(*,*) sum(m) ! executes on host
  write(*,*) gpusum(m) ! executes on device
end program
```

KERNELS

- Most Fortran kernels in CloverLeaf are doubly-nested loops over spatial indices
- Replace Fortran loops with global thread index calculation

- CloverLeaf is an explicit numerical method
  - Many kernel arguments read-only data
- Finite volume is low-order (small stencil)
  - Limited data reuse
- Use textures

EXPLICIT TEXTURE PROGRAMMING

```
module kernels
 real, pointer, texture :: bTex(:)
contains
 attributes(global) subroutine add(a,n)
  real :: a(*)
  integer, value :: n
  integer :: i
  i=(blockIdx%x-1)*blockDim%x+threadIdx%x
  if (i <= n) a(i) = a(i)+bTex(i)
 end subroutine add
end module kernels
```

```
program tex
  use kernels
  integer, parameter :: nb=1000, nt=256
  integer, parameter :: n = nb*nt
  real, device :: a_d(n)
  real, device, target :: b_d(n)
  real :: a(n)

  a_d = 1.0; b_d = 1.0

  bTex => b_d ! "bind" texture to b_d

  call add<<<nb,nt>>>(a_d,n)
  a = a_d
  if (all(a == 2.0)) print *, "OK"

  nullify(bTex) ! unbind texture
end program tex
```

IMPLICIT TEXTURES

- Declare kernel arguments as **intent(in)**
- Compiler will generate LDG instruction that loads data through texture path

```
module kernels
contains
  attributes(global) subroutine add(a,b,n)
    implicit none
    real :: a(*)
    real, intent(in) :: b(*)
    integer, value :: n
    integer :: i
    i=(blockIdx%x-1)*blockDim%x+threadIdx%x
    if (i <= n) a(i) = a(i)+b(i)
  end subroutine add
end module kernel
```

```
program ldg
  use kernels
  integer, parameter :: nb=1000, nt=256
  integer, parameter :: n = nb*nt
  real, device :: a_d(n), b_d(n)
  real :: a(n)

  a_d = 1.0; b_d = 1.0
  call add<<<nb,nt>>>(a_d, b_d, n)
  a = a_d
  if (all(a == 2.0)) print *, "OK"

end program lgd
```

IMPLICIT TEXTURES

- Verify
  - ```
    $ pgf90 -c -Mcuda=cc35,keepptx ldg.cuf
    $ grep ld.global.nc ldg.n001.ptx
       ld.global.nc.f32  %f1, [%rd10];
    ```
  - ```
    $ cuobjdump -sass ldg.o | grep LDG
            /*00f0*/  LDG.E R0, [R6];    /* 0x600210847f9c1801 */
    ```
- CUF kernels generate LDG when appropriate
- CC 3.5+

KERNELS

- Original code from ideal_gas_kernel

```
DO k=y_min,y_max
  DO j=x_min,x_max
    v=1.0_8/density(j,k)
    pressure(j,k)=(1.4_8-1.0_8)*density(j,k)*energy(j,k)
    pressurebyenergy=(1.4_8-1.0_8)*density(j,k)
    pressurebyvolume=-density(j,k)*pressure(j,k)
    sound_speed_squared=v*v*(pressure(j,k)*pressurebyenergy-pressurebyvolume)
    soundspeed(j,k)=SQRT(sound_speed_squared)
  ENDDO
ENDDO
```

GPU TECHNOLOGY CONFERENCE

KERNELS

- CUDA Fortran ideal_gas_kernel (base)

```
j = (blockIdx%x-1)*blockDim%x + threadIdx%x + x_min-1
k = (blockIdx%y-1)*blockDim%y + threadIdx%y + y_min-1

if (j <= x_max .and. k <= y_max) then
   v=1.0_8/density(j,k)
   pressure(j,k)=(1.4_8-1.0_8)*density(j,k)*energy(j,k)
   pressurebyenergy=(1.4_8-1.0_8)*density(j,k)
   pressurebyvolume=-density(j,k)*pressure(j,k)
   sound_speed_squared=v*v*(pressure(j,k)*pressurebyenergy-pressurebyvolume)
   soundspeed(j,k)=SQRT(sound_speed_squared)
end if
```

`density, energy` declared as `intent(in)`

KERNELS

- CUDA Fortran ideal_gas_kernel (opt)

```
j = (blockIdx%x-1)*blockDim%x + threadIdx%x + x_min-1
k = (blockIdx%y-1)*blockDim%y + threadIdx%y + y_min-1

if (j <= x_max .and. k <= y_max) then
   density_jk=density(j,k)
   v=1.0_8/density_jk
   pressure(j,k)=(1.4_8-1.0_8)*density_jk*energy(j,k)
   pressurebyenergy=(1.4_8-1.0_8)*density_jk
   pressurebyvolume=-density_jk*pressure(j,k)
   sound_speed_squared=v*v*(pressure(j,k)*pressurebyenergy-pressurebyvolume)
   soundspeed(j,k)=SQRT(sound_speed_squared)
end if
```

RESULTS

	Reported average time step per cell (10^-8 seconds) on K20c 2955 time steps				
Grid size	CUDA Fortran (base)	CUDA Fortran (opt)	CUDA C	OpenACC LOOPS	OpenACC KERNELS
960x960	1.57	1.43	1.59	2.19	2.05
1920x960	1.50	1.35	1.39	2.04	1.89
1920x1920	1.47	1.32	1.32	1.93	1.82
3840x1920	1.48	1.34	1.28	1.95	1.80
3840x3840	1.47	1.33	1.25	1.92	1.78

GPU TECHNOLOGY CONFERENCE

RESULTS

	Reported average time step per cell (10^-8 seconds) CUDA Fortran (opt)		
Grid size	K20c	K40m (base clocks)	K40m (boost clocks)
960x960	1.43	1.16	1.02
1920x960	1.35	1.09	0.96
1920x1920	1.32	1.06	0.93
3840x1920	1.34	1.06	0.93
3840x3840	1.33	1.06	0.92

GPU TECHNOLOGY CONFERENCE

SUMMARY

- New features result in more performance with less effort
  - Managed Memory — implicit data movement
  - CUF Kernels/reduction intrinsics — implicit kernel generation
  - `intent(in)` kernel arguments — implicit textures

GPU TECHNOLOGY CONFERENCE

THANK YOU

JOIN THE CONVERSATION

#GTC15