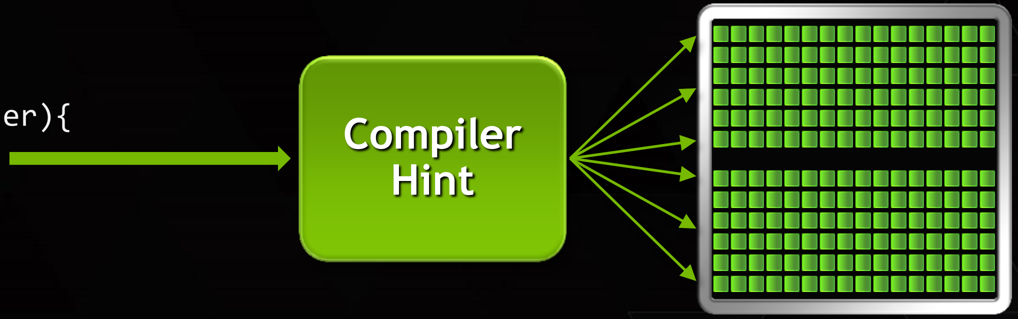
GPU TECHNOLOGY CONFERENCE

C++ CLASS MANAGEMENT WITH OPENACC 2.0

MATHEW COLGROVE PGI/NVIDIA DEV TECH ENGINEER

OPENACC: OPEN, SIMPLE, PORTABLE

```
#pragma acc data \
        copy(b[0:n][0:m]) \
        create(a[0:n][0: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][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];
}}
```



- ✓ Open Standard
- ✓ Straightforward, Compiler-Driven Approach
- ✓ Performance Portable Across Platforms

OPENACC AND C++

- PGI is actively pushing forward OpenACC and C++ interoperability
- Since GTC 2014, PGI has added support for:
  - C++ Classes, “this” pointers, Data Members, Class Methods
  - Templates, Lambdas
  - Full compatibility with g++ through GNU 4.9
- Work continues: PGI is collaborating with Sandia to define and drive forward interoperability of C++ and OpenACC

OPENACC AND C++

```
#include "vtype.h"
int main () {
   long n=1024;
   vtype<float> x(n), y(n);
   x.init(1.0,1.0);
   y.init(2.0,2.0);

#pragma acc parallel loop
    for (int i = 0; i < x.size(); ++i) x[i] += y[i];
    // size method, [] operator implicitly compiled for device

#ifdef _OPENACC
    x.update_host();
    y.update_host();
    // Need to add methods to perform data movement
#endif
```

UNSTRUCTURED DATA LIFETIMES

```
#include <iostream>

template<typename T> class vtype {
    long _size;
    T* _data;
public:
    explicit vtype(long size) : _size(size) {
        _data = new T[_size];
              // Copy the 'this' pointer and shallow copy of data members
        #pragma acc enter data copyin(this)
              // Create the _data vector on device and 'attach' to the class
        #pragma acc enter data create(_data[0:_size])
    }
    ~vtype() {
        delete [] _data;
              // Delete the device data
        #pragma acc exit data delete(_data,this)
    }
```

OPENACC AND C++ CLASS METHODS

```
long size() { return _size; }
inline T& operator[](long i) const { return _data[i]; }

#pragma acc routine seq
T initValues(T start, T inc, long val) {
     return start+(inc*val);
}
void init(T start, T inc) {
    #pragma acc parallel loop gang vector present(_data[0:_size])
    for (long i = 0; i < _size; ++i) {
        _data[i] = initValues(start, inc, i);
    }
}
void update_device(){
    #pragma acc update device(_data[0:_size])
}
void update_host(){
    #pragma acc update host(_data[0:_size])
}
```

DATA MANAGEMENT—DEEP UPDATE

- What happens if “_data” array isn’t a fundamental type but another class?
  - If the class contains no dynamic data members, then treat it as a fundamental data type
  - If the class contains dynamic data members, recursively call “update”

```
void update_device() {
   for (int i=0; i < _size; ++i) {
       _data[i]->update_device();
   }
}
 void update_host() {
   for (int i=0; i < _size; ++i) {
       _data[i]->update_host();
   }
}
```

OPENACC ROUTINE DIRECTIVE

- Specify functions to be compiled for device execution
- Clauses define the type of parallel loop in which the function will be called: gang, worker, vector, seq

```
#pragma acc parallel loop gang \
            vector_length(VL)
   for(int i=0;i<N;i++)
    fun_vec(…);
}
```

```
#pragma acc routine vector
void fun_vec(…) {
  #pragma acc loop vector
  for(int i=0;i<N;i++)
    fun_seq(…);
}
```

```
#pragma acc routine seq
void fun_seq(…) {
  #pragma acc loop seq
  for(int i=0;i<N;i++)
    ...
}
```

OPENACC ROUTINE

- PGI C++ automatically compiles visible class methods for device execution if an instance of the class is referenced in an OpenACC region
- Critical for support of Templates and Lambdas
- PGI is advocating to add this behavior to the OpenACC specification

```
T1 &vtype<T1>::operator [](long) const [with T1=float]:
      1, include "vtype.h"
          24, Generating implicit acc routine seq
              Generating Tesla code

T1 vtype<T1>::initValues(T1, T1, long) [with T1=float]:
      1, include "vtype.h"
          25, Generating implicit acc routine seq
              Generating Tesla
```

EXAMPLES

- Template Container class
  1. Data is a simple scalar type
  2. Data is a simple class
  3. Data is a class with dynamic single-dimensional data members
  4. Data is a class with dynamic multi-dimensional data members allocated via a template class

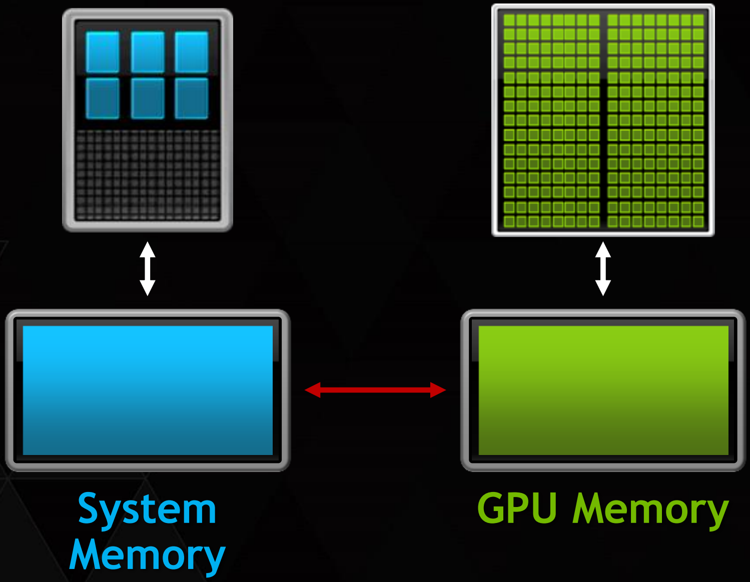
Available at: www.pgroup.com/lit/samples/gtc15_s5233.tar

AGGREGATE DATA TYPES

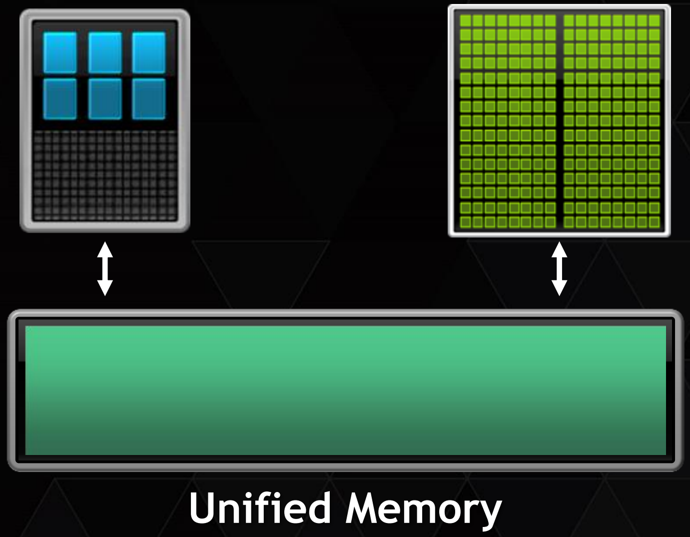
- Aggregate data types with dynamic data members
  - Not part of OpenACC 2.0 Specification
  - Currently up to the user to build and update device side structures
  - What about STL containers such as Vector or Map?
  - Technical report from Nov 2014 attempts to address these limitations
    - www.openacc.org/sites/default/files/TR-14-1.pdf
- It’s a very difficult problem to solve!
- And one that may soon be moot …

CUDA Unified Memory

Developer View Today

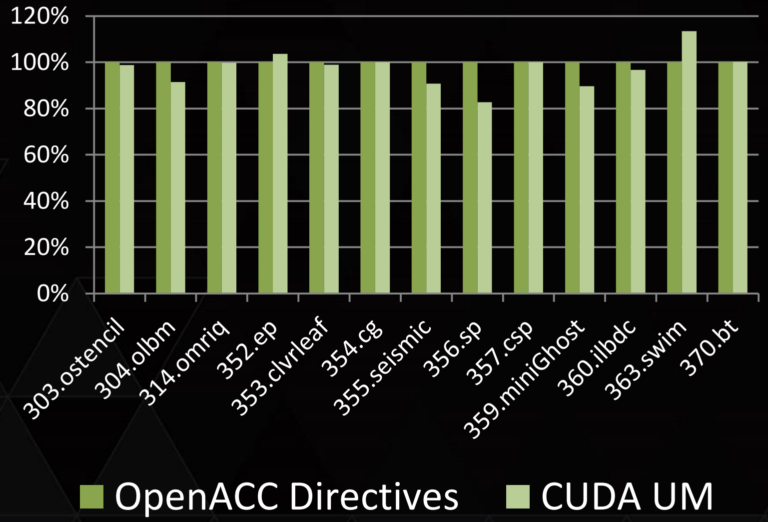


Developer View With Unified Memory



OPENACC AND CUDA UNIFIED MEMORY

OpenACC directive-based data movement vs OpenACC w/CUDA 6.5 Unified Memory on Kepler. (PGI 15.3 Beta)



Features:

- Fortran ALLOCATE and C/C++ malloc/calloc/new can automatically use CUDA UM
- No explicit transfers needed for dynamic data

Limitations:

- Supported only for dynamic data
- Program dynamic memory size is limited by UM data size
- UM data motion is synchronous
- Can be unsafe

PGI C++ OPENACC - UNIFIED MEMORY

```
$ pgc++ -ta=tesla:managed ...

vtype(long size) : _size(size) {
    _data = new T[_size];
    // Copy the 'this' pointer and shallow copy of data members
    #pragma acc enter data copyin(this)
    // Dynamic data managed by Unified Memory
}
~vtype() {
    delete [] _data;
    // Delete the device data
    #pragma acc exit data delete(this)
}

// The update methods are no longer needed
```

EXAMPLES

- Let's revisit the same examples, but this time simplified for use with Unified Memory

1. Data is a simple scalar type
2. Data is a simple class
3. Data is a class with dynamic single dimension data members
4. Data is a class with dynamic multi-dimensions data members allocated via a template class

Available at: www.pgroup.com/lit/samples/gtc15_s5233.tar

GPU TECHNOLOGY CONFERENCE

FUTURE WORK

- Exception handling
- Function pointers and virtual functions
- STL Container Types
- STL Algorithms
- C++17 parallel for_each
- Performance Tuning
- Others?

SUMMARY

- C++ support in OpenACC is maturing rapidly
  - Unstructured data regions
  - Routine directives
  - “this” pointers
  - automatic “routines”
- CUDA Unified Memory promises to simplify managing deep data constructs
- PGI is actively working to improve C++ support in OpenACC

MORE OPENACC SESSIONS AT GTC

S5160	Experiences in Porting Scientific Applications to GPUs	Thu 1400-1450	220C
S5202	Porting Computational Physics Applications to the Titan Supercomputer with OpenACC and OpenMP	Thu 1500-1525	220C
S5382	OpenACC 2.5 and Beyond	Thu 1530-1555	220C
S5322	Accelerating CICE on the GPU	Thu 1700-1725	210F
	OpenACC Hang-out	Thu 1700-1800	Pod C
S5195	Advanced OpenACC Programming	Fri 0900-1020	210C
S5340	OpenACC and C++: An Application Perspective	Fri 1030-1055	210C
S5531	The RAMSES Code for Numerical Astrophysics	Fri 1030-1055	210D
S5198	Panel on GPU Computing with OpenACC and OpenMP	Fri 1100-1150	210C

GPU TECHNOLOGY CONFERENCE

EMAIL: MCOLGROVE@NVIDIA.COM

TWITTER: @PGICOMPILERS

JOIN THE CONVERSATION

#GTC15