OpenACC 2.0 and Beyond

PGI Accelerator Compilers and Tools

PGI®

One Slide Intro to OpenACC Directives

Manage Data Movement

Initiate Parallel Execution

Optimize Loop Mappings

```
#pragma acc data copyin(x[0:n],y[0:n]) copyout(z[0:n])
{
  ...
  #pragma acc parallel
  {
    #pragma acc loop gang vector
    for (i = 0; i < n; ++i) {
      z[i] = x[i] + y[i];
      ...
    }
  }
  ...
}
```

- Incremental
- Single source
- Interoperable
- Performance portable
- CPU, GPU, MIC

OpenACC
Directives for Accelerators

PGI®

OpenACC 2.0

Highlights

- acc routine
- acc enter / exit data
- acc atomic
- acc wait async
- acc parallel wait()
- acc loop tile
- default(none)
- device_type(...)
- new API routines
- many clarifications

PGI®

OpenACC 2.0

acc routine

```
#pragma acc routine vector
float dotprod( float* a, float* b, int n ){
    float sum = 0.0f;
    #pragma acc loop vector reduction(+:sum)
    for( int i = 0; i < n; ++ i
        sum += a[i]*b[i];
    return sum;
}
```

PGI®

OpenACC 2.0

acc enter data and acc exit data

```
template<typename T>class v{
    T* _data;  size_t _size; ...
    move_to_device(){
        #pragma acc enter data copyin(this, \
                                  _data[0:_size])
    }
    update_host(){
        #pragma acc update self(_data[0:_size])
    }...
```

PGI®

OpenACC 2.0

acc atomic

```
#pragma acc parallel loop
for( i = 0; i < n; ++i ){
    x = index[i];
    #pragma acc atomic update
    hist[x]++;
}
```

PGI®

OpenACC 2.0

acc wait async

```
#pragma acc parallel loop async(1)
for(...){...}
#pragma acc parallel loop async(2)
for(...){...}
#pragma acc wait(1) async(2)
#pragma acc parallel loop async(2) wait(1)
for(...){...}
```

PGI®

PGI 2015 Additions

C++ class data member in OpenACC data clauses

```
template<typename T> class myvect{
    T* _data;  size_t _size;
public: // ...
    void dev_create(){
       #pragma acc enter data copyin(this)
       #pragma acc enter data copyin(_data[0:_size])
    }
    void host_update(){
       #pragma acc update self(_data[0:_size])
    }
```

PGI®

PGI 2015 Additions

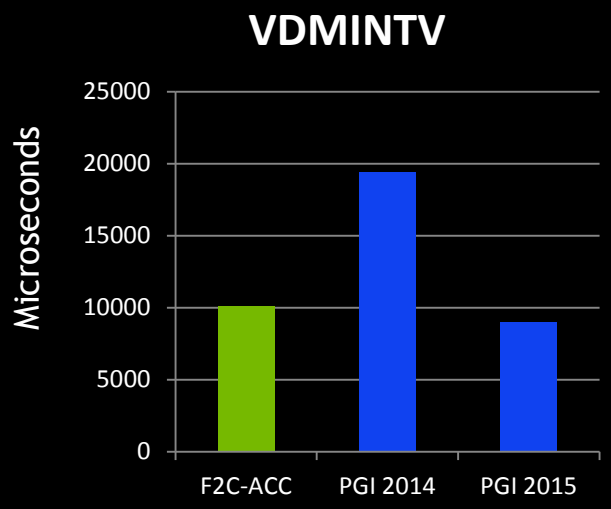
Managed Memory Support (beta feature)

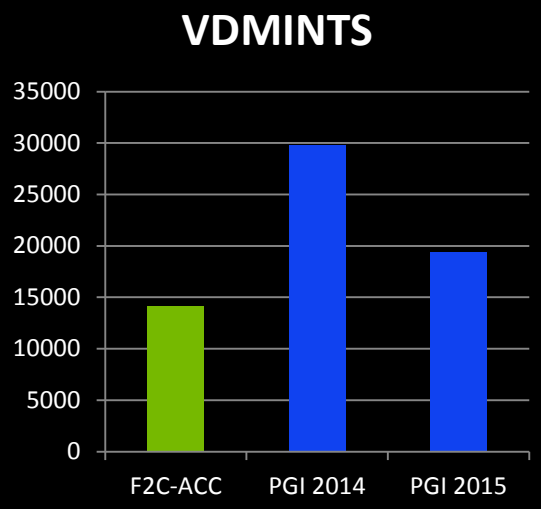
- pgc++ -ta=tesla:managed
- malloc, calloc, free, new, delete, allocatable replaced with managed allocate/free
- limitations

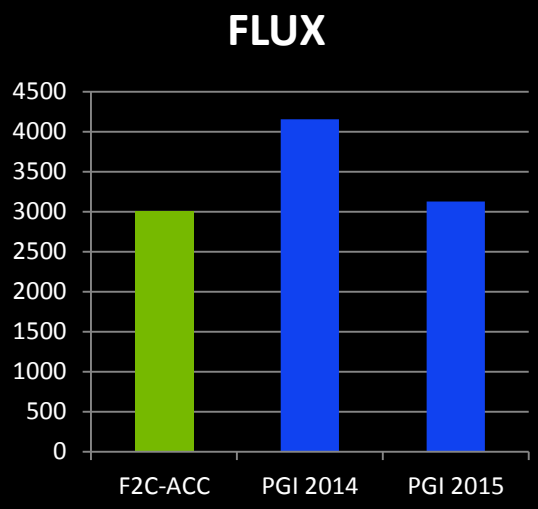
PGI®

PGI 2015 OpenACC Performance – NIM (NOAA)

All times measured on a K20x not including data transfers from host memory to device memory







PGI 2015 OpenACC Performance Enhancements:

- ✓ !$ACC CACHE directive
- ✓ Scalar replacement optimizations
- ✓ Variable length VECTOR support
- ✓ Short loop optimizations

PGI®

OpenACC 2.5 (in design)

acc data copy(x) == present_or_copy(x)

```
#pragma acc data present_or_copy(x[0:n]) copy(b[0:n])
{....}
```

PGI®

OpenACC 2.5 (in design)

acc declare(allocatable)

```
module m
    real, allocatable :: a(:,:)
    !$acc declare create(a)
end module
subroutine init(n)
    use m
    allocate(a(n,n))
    ...
```

PGI®

OpenACC 2.5 (in design)

default(present)

```
#pragma acc parallel loop default(present)
for( i = 0; i < n; ++i )
    a[i] = fexpf(b[i]) * cosf(c[i]);
```

PGI®

OpenACC 3.0 (in design)

Deep Copy – Data Structure Management

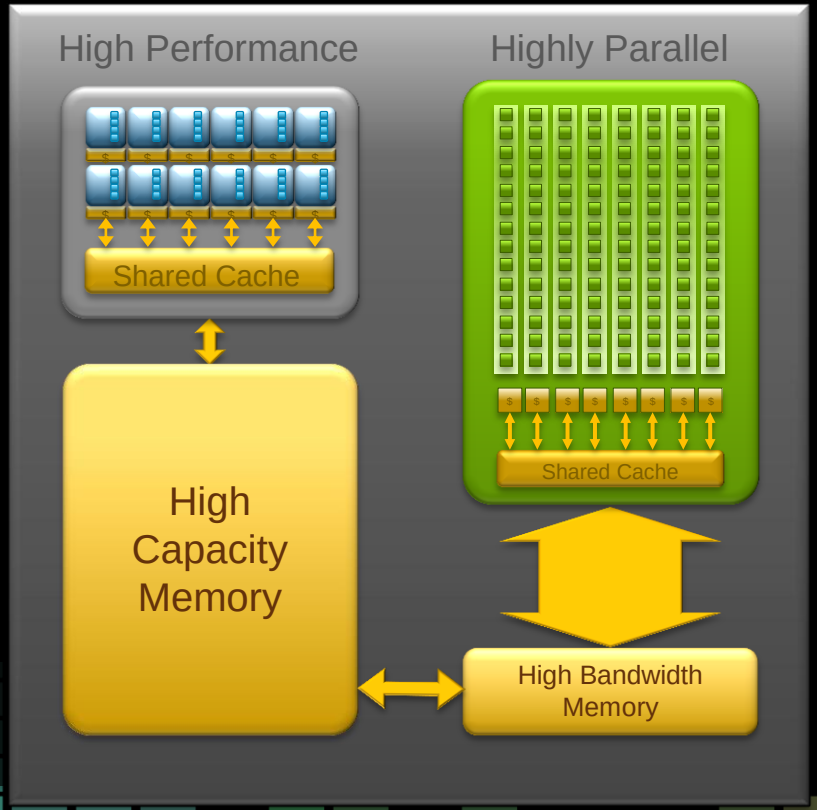
```
template<typename T>class reactor{
    class magnet* m;
    class laser* l;
    class coolant* c;
    class steampipe* s;
};
...
class reactor R;
#pragma acc enter data copyin(R)
```

PGI®

Future of OpenACC

On Future Supercomputers

- Descriptive
- Performance Portable
- Data Management
- Parallelism Management
- PGI Commitment



PGI®