ICHEC
Irish Centre for High-End Computing

Code acceleration with HMPP

Outcomes from HMPP training

HMPP in a nutshell

- Language extension for hardware accelerators
  - For C, Fortran and C++ soon
  - Based on compiler directives
  - Easy to learn and to use
- What OpenMP is for multi-thread programming

HMPP rational

- Hardware accelerators are hard to program
  - Mostly limited to C API and/or C extensions
  - Low-level programming
  - Hard to tune and to debug
  - Nightmare to maintain
- What about portability?
  - Development environment
  - Hardware

ICHEC
Irish Centre for High-End Computing

HMPP answer

- Compiler directives
  - No code “modifications”, just comments if not recognised by the compiler
  - Mostly hardware independent
  - Can address different targets and strategies
- Run time environment
  - Low-level optimisations undertaken by HMPP itself
  - Always a fallback possibility to pure CPU code

HMPP targets

- Current:
  - CUDA for Nvidia GPU
  - CAL/IL or BROOK for ATI/AMD GPU
  - C for debugging purpose
  - SSE for SSE vectorisation
  - CELL for IBM Cell processors (limited support)
- Future:
  - OPENCL for even more portability
  - …

HMPP basic: codelet/callsite

- Paired directives
  - codelet: routine implementation
  - callsite: routine invocation
- Unique label for referencing them
- 1 for 1 association in the code
  - As many individual codelet (re)definitions as actual callsite invocations

HMPP codelet example

```
#pragma hmpp label1 codelet, args[B].io=out, args
   [C].io=inout, target=CUDA:CAL/IL
void myFunc(int n, int A[n], int B[n], int C[n])
{
   for(int i=0 ; i<n ; i++)
   {
      B[i] = A[i] * A[i];
      C[i] = C[i] * A[i];
   }
}
```

HMPP callsite example

```
for(int i=0 ; i<n ; i++)
    A[i] = C[i] = i;

for(int i=0 ; i<n ; i++)
{
#pragma hmpp label1 callsite
    myFunc(n, A, B, C);
}
```

More HMPP features

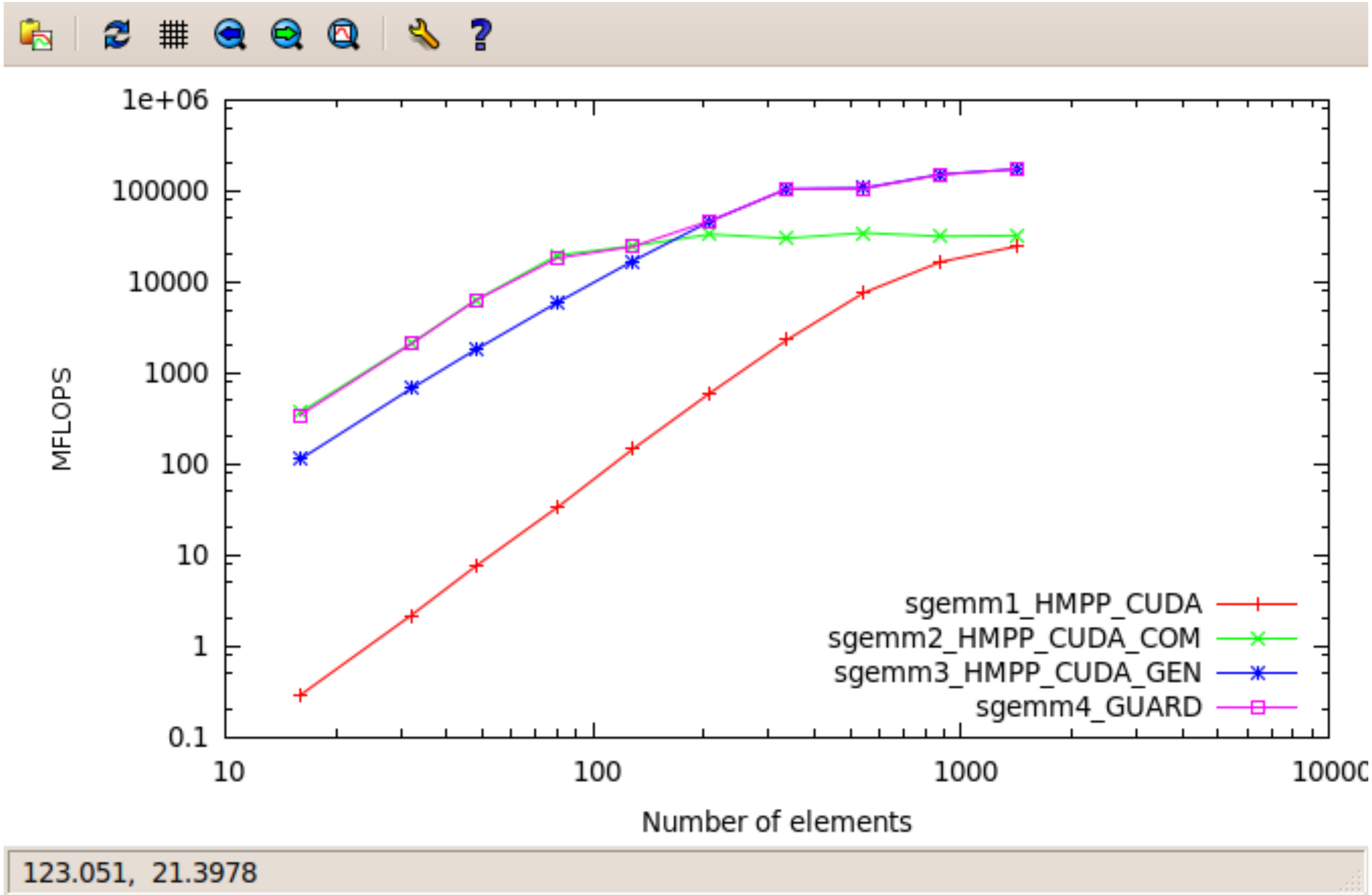
- Hardware management
  - allocate: reserve hardware and allocate memory
  - release: opposite actions
- Data transfer management
  - advanceload: explicit host to device transfer
  - delegatestore: explicit device to host transfer

HMPP allocate/release example

```
for(int i=0 ; i<n ; i++)
    A[i] = C[i] = i;
#pragma hmpp label1 allocate
for(int i=0 ; i<n ; i++)
{
#pragma hmpp label1 callsite
    myFunc(n, A, B, C);
}
#pragma hmpp label1 release
```

Even more features

- Tones of memory management options
- Asynchronous data transfers
- Thread synchronisation
- Codelet grouping
- Conditional invocation
- Advanced algorithmic optimisations
  - Loop parallelisation
  - Loop unrolling / jamming


1e+06
100000
10000
1000
100
10
1
0.1
MFLOPS
10
100
1000
10000
Number of elements
sgemm1_HMPP_CUDA
sgemm2_HMPP_CUDA_COM
sgemm3_HMPP_CUDA_GEN
sgemm4_GUARD
123.051, 21.3978

Conclusion

- Easy to develop / maintain
- Efficient
- Cyclic approach to hardware acceleration
- Hardware-portable
- Possibly software-portable