The Role of Standards in Heterogeneous Programming

Multi-core Challenge
Bristol UWE

Paul Keir - Codeplay Software Ltd.

45 York Place, Edinburgh EH1 3HP

June 12th, 2013

Codeplay Software Ltd.

- Incorporated in 1999
- Based in Edinburgh, Scotland
- 30 full-time employees
- Compilers, optimisation and language development
  - GPU and Heterogeneous Architectures
  - Increasingly Mobile and Embedded CPU/GPU SoCs
- Commercial partners:
  - Qualcomm, Movidius, AGEIA, Fixstars
  - Many other partners remain confidential
- Member of two 3-year EU FP7 research projects:
  - Peppher and LPGPU
- Sony-licensed PS3^{TM} middleware provider
- Contributing member of Khronos group since 2006
  - Working towards latest version of OpenCL

C++ AMP

- Microsoft-backed open standard for heterogeneous compute
- Announced June 2011
- Single-source programming model
- GPU step-debugging with Visual Studio IDE
- Use of lambdas emphasises C++11 support
- Two C++ language extensions:
  - Function qualifier: `restrict`
  - Storage class: `tile_static`
- Non-Microsoft implementations?
  - Shevlin Park - Intel prototype
  - Clang patch from Dave McFarland

C++ AMP Restrictions - Selected Highlights

- No `volatile` members or variables
- No support for `char` type
- No pointers to pointers
- No `virtual` base class or member functions
- No function pointers
- No exceptions
- No recursion
- No `goto` statements

C++ AMP Example - Vector Addition

```
void vec_add(int n, int *pA, int *pB, int *pC)
{
  array_view<int       > a(n,pA);
  array_view<const int> b(n,pB), c(n,pC);
  parallel_for_each(a.extent,
    [&](index<1> i) restrict(amp)
  {
    a[i] = b[i] + c[i];
  });
}
```

- Second argument to `parallel_for_each` is a C++ lambda
  - May be any suitable function object i.e. One with operator:
  - `template <int N> operator()(index<N>)`
- The `operator[]` of `array_view` is overloaded
  - For: `template <int N> index<N>`

C++11 Highlights

- Lambda expressions
- `auto`-typed variables
- Declared type of an expression (`decltype`)
- Static assertions (`static_assert`)
- Variadic templates
- Alias templates
- Generalized constant expressions (`constexpr`)

C++14 Committee Draft (CD) in Public Review

```
auto Identity = [](auto a) { return a; };
```

```
for_each(begin(v), end(v), [](auto &x) {
  cout << x;
});
```

- Generic (polymorphic) lambda expressions
  - Support for auto as a type name in a lambda

C++14 Committee Draft (CD) in Public Review

```
auto sum(int i) {
  if (i == 0)
    return i;
  else
    return i + sum(i-1);
}
```

- Return type deduction for normal functions
  - Non-lambdas can now use `auto` and a trailing return type
  - Restriction on a single return statement relaxed for both

OpenMP 4.0

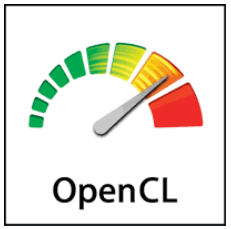
- Cross-platform standard for shared memory parallelism
- Traditionally popular in High Performance Computing (HPC)
- A single-source approach for C, C++ and Fortran
- Makes essential use of compiler pragmas
- Supported by most modern compilers
  - OpenMP in Clang development presented April in Paris
  - Intel OpenMP runtime now BSD licensed: `openmprtl.org`
- Release Candidate 2 now available for public comments
- Summer release target for OpenMP 4.0
- Significant new features include:
  - Improved SIMD support
  - User defined reductions
  - Support for accelerators

OpenMP 4.0 - Vector Addition on GPU

```
void vec_add(int n, float *pA,
             float const *pB, float const *pC)
{
#pragma omp target map(from:pB[0:n],pC[0:n]) \
                   map(to:pA[0:n])
  {
#pragma omp teams
#pragma omp distribute
#pragma omp parallel for
    for (int i = 0; i < n; ++i)
      pA[i] = pB[i] + pC[i];
  }
}
```

OpenCL (Open Computing Language)

- Royalty-free, cross-platform standard governed by Khronos
- Portable parallel programming of heterogeneous systems
- Memory and execution model similar to CUDA
- OpenCL C kernel language based on ISO C99 standard
  - Source distributed with each application
  - Kernel language files compiled at runtime




```
kernel void vec_add(global       float *a,
                    global const float *b,
                    global const float *c) {
  int id = get_global_id(0);
  a[id] = b[id] + c[id];
}
```

OpenCL Working Groups

- OpenCL HLM
  - C/C++ syntax/compiler extensions
  - A High Level Model (HLM) for OpenCL
  - Working group chaired by Codeplay's Andrew Richards

OpenCL Working Groups

- OpenCL HLM
  - C/C++ syntax/compiler extensions
  - A High Level Model (HLM) for OpenCL
  - Working group chaired by Codeplay's Andrew Richards
- OpenCL SPIR
  - Standard Portable Intermediate Representation (SPIR)
  - A portable LLVM-based compiler IR
  - A binary distribution format: don't ship source
  - Faster compile times
  - Provisional specification version 1.0 public
  - Host API defined in OpenCL Extension Specification 1.2
    - `llvm-as foo.ll -o foo.bc`
    - Host calls `clCreateProgramFromBinary`

OpenCL Working Groups

- OpenCL HLM
  - C/C++ syntax/compiler extensions
  - A High Level Model (HLM) for OpenCL
  - Working group chaired by Codeplay's Andrew Richards
- OpenCL SPIR
  - Standard Portable Intermediate Representation (SPIR)
  - A portable LLVM-based compiler IR
  - A binary distribution format: don't ship source
  - Faster compile times
  - Provisional specification version 1.0 public
  - Host API defined in OpenCL Extension Specification 1.2
    - `llvm-as foo.ll -o foo.bc`
    - Host calls `clCreateProgramFromBinary`
- One of many new low-level virtual heterogeneous ISAs
  - Proprietary: NVPTX and AMDIL
  - Cross-platform: HSAIL
    - HSA Programmer's Reference Manual 0.95 now available
  - A new language war!

Vector Addition using OpenCL SPIR (1 of 3)

```
define spirkrnl void @vec_add(
  float addrspace(1)* nocapture %a,
  float addrspace(1)* nocapture %b,
  float addrspace(1)* nocapture %c) nounwind
{
  %1 = call i32 @get_global_id(i32 0)

  %2 = getelementptr float addrspace(1)* %a, i32 %1
  %3 = getelementptr float addrspace(1)* %b, i32 %1
  %4 = getelementptr float addrspace(1)* %c, i32 %1

  %5 = load float addrspace(1)* %3, align 4
  %6 = load float addrspace(1)* %4, align 4
  %7 = fadd float %5, %6

  store float %7, float addrspace(1)* %2, align 4
  ret void
}
```

Vector Addition using OpenCL SPIR - Header (2 of 3)

```
target datalayout = "i1:8-i8:8-i16:16-i32:32-i64:64-f32:32-f6
target triple = "spir"
declare i32 @get_global_id(i32)

define spirkrnl void @vec_add(
  float addrspace(1)* nocapture %a,
  float addrspace(1)* nocapture %b,
  float addrspace(1)* nocapture %c) nounwind
{
  %1 = call i32 @get_global_id(i32 0)
  %2 = getelementptr float addrspace(1)* %a, i32 %1
  %3 = getelementptr float addrspace(1)* %b, i32 %1
  %4 = getelementptr float addrspace(1)* %c, i32 %1
  %5 = load float addrspace(1)* %3, align 4
  %6 = load float addrspace(1)* %4, align 4
  %7 = fadd float %5, %6
  store float %7, float addrspace(1)* %2, align 4
  ret void
}
```

Vector Addition using OpenCL SPIR - Metadata (3 of 3)

```
!opencl.kernels = !{!0}

!0 = metadata !{
  void (float addrspace(1)*, float addrspace(1)*,
        float addrspace(1)*)* @vec_add, metadata !1,
  metadata !2, metadata !3, metadata !4, metadata !5}
!1 = metadata !{metadata !"address_qualifier",
                i32 1, i32 1, i32 1}
!2 = metadata !{metadata !"access_qualifier",
                i32 1, i32 0, i32 0}
!3 = metadata !{metadata !"arg_type_name",
  metadata !"float*",
  metadata !"float*",
  metadata !"float*"}
!4 = metadata !{metadata !"arg_type_qualifier",
  metadata !"", metadata !"", metadata !""}
!5 = metadata !{metadata !"arg_name",
  metadata !"a", metadata !"b", metadata !"c"}
```

Summary

- C++ AMP 1.0 specification available
  - Microsoft implementation included with Visual Studio 2012
- C++14 Committee Draft (CD) in Public Review
- OpenMP 4.0 this summer
  - Release Candidate 2 now available for public comments
- The next OpenCL
  - OpenCL HLM
    - C/C++ syntax/compiler extensions
  - OpenCL SPIR
    - Provisional 1.0 specification on Khronos website
- HSA
  - Programmer's Reference Manual (HSAIL) now available
  - Includes formal EBNF grammar

Summary

- C++ AMP 1.0 specification available
  - Microsoft implementation included with Visual Studio 2012
- C++14 Committee Draft (CD) in Public Review
- OpenMP 4.0 this summer
  - Release Candidate 2 now available for public comments
- The next OpenCL
  - OpenCL HLM
    - C/C++ syntax/compiler extensions
  - OpenCL SPIR
    - Provisional 1.0 specification on Khronos website
- HSA
  - Programmer's Reference Manual (HSAIL) now available
  - Includes formal EBNF grammar
- Future standard releases
  - Fortran 201x, OpenACC 2.0