Programação Paralela: das *threads* aos FPGAs

C++ Single-source Heterogeneous Programming for Acceleration Offload (SYCL)

Prof. Ricardo Menotti
menotti@ufscar.br

Prof. Maurício Acconcia Dias
macccdias@gmail.com

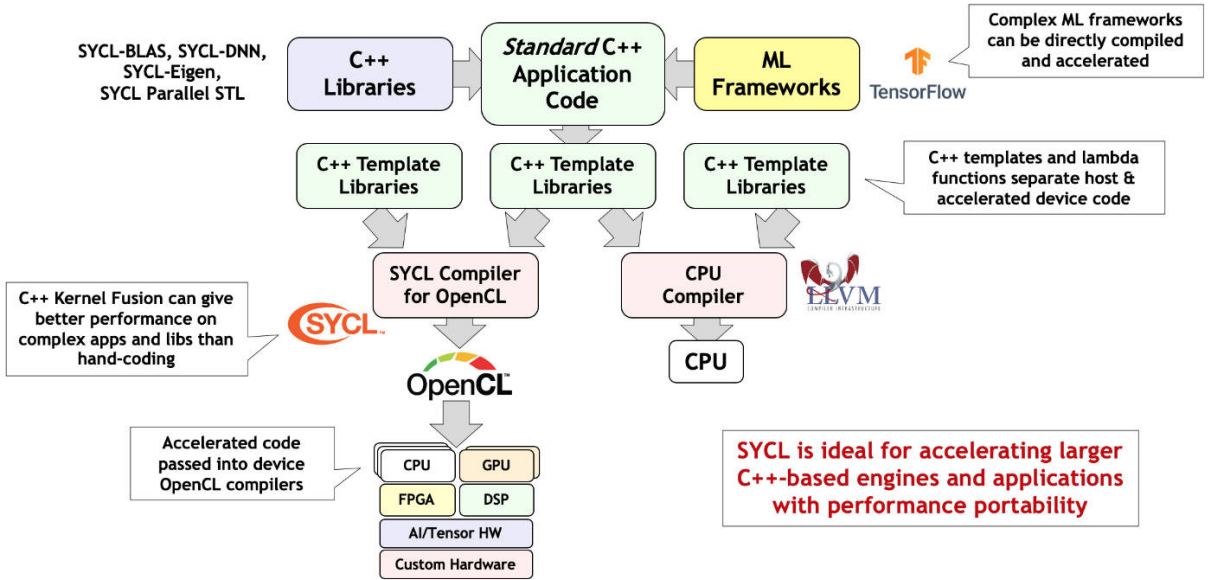
Prof. Helio Crestana Guardia
helio.guardia@ufscar.br

Departamento de Computação
Universidade Federal de São Carlos

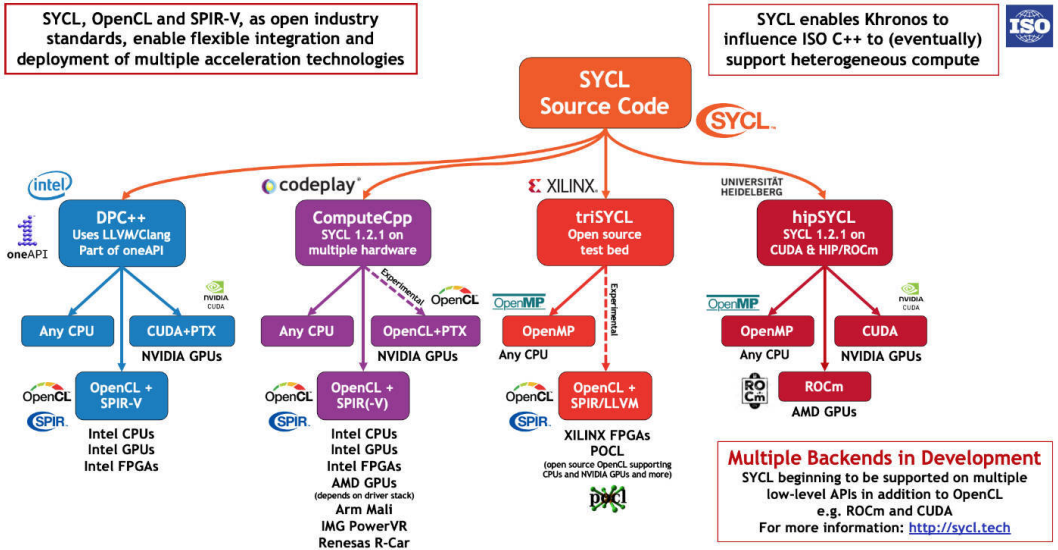
Atualizado em: 13 de junho de 2020

Conteúdo

Ecosistema


SYCL-BLAS, SYCL-DNN,
SYCL-Eigen,
SYCL Parallel STL
C++ Libraries
Standard C++ Application Code
ML Frameworks
TensorFlow
Complex ML frameworks can be directly compiled and accelerated
C++ Template Libraries
C++ Template Libraries
C++ Template Libraries
C++ templates and lambda functions separate host & accelerated device code
SYCL Compiler for OpenCL
CPU Compiler
LLVM
C++ Kernel Fusion can give better performance on complex apps and libs than hand-coding
SYCL
CPU
OpenCL
Accelerated code passed into device OpenCL compilers
CPU
GPU
FPGA
DSP
AI/Tensor HW
Custom Hardware
SYCL is ideal for accelerating larger C++-based engines and applications with performance portability

Implementações



C++ moderno [11, 14, 17, 20]

- Mais ‘restrito’ do que C (tipos, constantes etc.)
- Programação genérica (*templates*)
- Funções anônimas/oclusão (*lambdas*)
- Inferência de tipos na inicialização (`auto`)

Revisão C++

Mais ‘restrito’ do que C (tipos, constantes etc.)

```
1  void g1(std::string& s);
2
3  void f1(const std::string& s)
4  {
5    g1(s);           // Compile-time Error since s is const
6    std::string localCopy = s;
7    g1(localCopy);  // Okay since localCopy is not const
8  }
```

Revisão C++

Programação genérica (*templates*)

```
1  template <typename T>
2  class vec3 {
3      T val[3];
4      public:
5          vec3 (T x, T y, T z) {
6              val[0] = x;
7              val[1] = y;
8              val[2] = z;
9          }
10 };
11 vec3<double> a(1.0, 2.0, 3.0);
12 vec3<int> b(1, 2, 3);
```

Revisão C++

Funções anônimas/oclusão (*lambdas*)

- Cria uma função anônima que pode capturar variáveis do escopo
  - [] não captura nada
  - [&] captura por referência
  - [=] captura por valor (cópia)
  - [a, &b] captura variáveis explicitamente

```
1  void func (std::function<void(int)> f) { int x = 6; f(x); }
2
3  {
4     int a = 6;
5     func([&](int q){ if (q == a) std::cout << "success"; });
6  }
```

Revisão C++

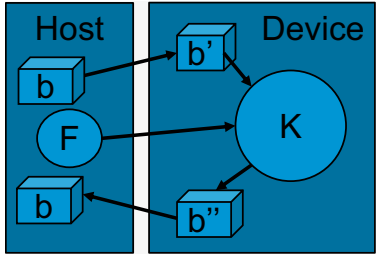
Inferência de tipos na inicialização

```
1  std::map<std::pair<std::string,std::string>,std::vector<int>>::iterator iter;
2  iter = mymap.begin();
3
4  auto iter = mymap.begin();
```

Linguagem

Pseudo-código

```
1  selector = default_selector()
2  q = queue(selector)
3  b = buffer (double, 1000)
4  q.submit (
5      F() {
6          a = accessor(b, READ_AND_WRITE)
7          K (i) {
8              a[i] = a[i] * 2
9          }
10     }
11 )
12 q.wait_and_throw()
13 a2 = accessor(b, READ_ONLY) printf("%f", a2[0])
```



Kernels

- Três tipos de kernels:
  - `single_task`: roda um instância do kernel;
  - `parallel_for`: roda várias instâncias, baseado no intervalo definido;
  - `parallel_for_work_[group|item]`: roda várias instâncias, permite paralelismo herárquico

Operadores de intervalo (*range*)

- SYCL possui sete classes para decomposiçao de dados:
  - **`range, nd_range, id, item, nd_item, group, h_item`**
- **`range<dimensions>(<size of dimension>)`**
  - `range<1>(128)`
  - `range<2>(640, 480)`
- **`id<dimensions>`** - fornece o índice no intervalo
  - `id<1>a; size_t index = a[0];`
  - `id<2>b; size_t x = b[0]; size_t y = b[1];`
- **`nd_range<dimensions>(range<dimension> global_size, range<dimension> local_size)`**
  - `nd_range<1>(range<1>(256),range<1>(128))`
- **`nd_item<dimensions>`** - fornece o índice no intervalo, com mais funcionalidades
  - `nd_item<1>a;`
  - `size_t global_index = a.get_global_id(0);`
  - `size_t local_index = a.get_local_id(0);`

Linguagem

Hello, World! (1/3)

```
1  #include <iostream>
2  #include <CL/sycl.hpp>
3
4  class vector_addition;
5
6  int main(int, char**) {
7     cl::sycl::float4 a = { 1.1, 2.2, 3.3, 4.4 };
8     cl::sycl::float4 b = { 4.4, 3.3, 2.2, 1.1 };
9     cl::sycl::float4 c = { 0.0, 0.0, 0.0, 0.0 };
10
11    cl::sycl::default_selector device_selector;
12
13    cl::sycl::queue queue(device_selector);
14    std::cout << "Running on "
15              << queue.get_device().get_info<cl::sycl::info::device::name>()
16              << "\n";
```

Linguagem

Hello, World! (2/3)

```
17 {
18   cl::sycl::buffer<cl::sycl::float4, 1> a_sycl(&a, cl::sycl::range<1>(1));
19   cl::sycl::buffer<cl::sycl::float4, 1> b_sycl(&b, cl::sycl::range<1>(1));
20   cl::sycl::buffer<cl::sycl::float4, 1> c_sycl(&c, cl::sycl::range<1>(1));
21
22   queue.submit([&] (cl::sycl::handler& cgh) {
23      auto a_acc = a_sycl.get_access<cl::sycl::access::mode::read>(cgh);
24      auto b_acc = b_sycl.get_access<cl::sycl::access::mode::read>(cgh);
25      auto c_acc = c_sycl.get_access<cl::sycl::access::mode::discard_write>(cgh);
26
27      cgh.single_task<class vector_addition>([=] () {
28      c_acc[0] = a_acc[0] + b_acc[0];
29      });
30   });
31 }
```

Linguagem

Hello, World! (3/3)

```
32      std::cout << "  A { " << a.x() << ", " << a.y() << ", "
33                << a.z() << ", " << a.w() << " }\n"
34                << "+ B { " << b.x() << ", " << b.y() << ", "
35                << b.z() << ", " << b.w() << " }\n"
36                << "------------------\n"
37                << "= C { " << c.x() << ", " << c.y() << ", "
38                << c.z() << ", " << c.w() << " }"
39                << std::endl;
40
41      return 0;
42  }
```

Bibliografia

- SYCL Overview - The Khronos Group Inc
- Intel(r) oneAPI Toolkits (Beta)
- SYCL and DPC++ for Aurora
- Training examples for SYCL
- ComputeCpp Guide
- Introdução ao SYCL
- Deep C (and C++)
- Referência C++

Programação Paralela: das *threads* aos FPGAs

C++ Single-source Heterogeneous Programming for Acceleration Offload (SYCL)

Prof. Ricardo Menotti
menotti@ufscar.br

Prof. Maurício Acconcia Dias
macccdias@gmail.com

Prof. Helio Crestana Guardia
helio.guardia@ufscar.br

Departamento de Computação
Universidade Federal de São Carlos

Atualizado em: 13 de junho de 2020