OneAPI for heterogeneous computing

VINCENZ DUMANN

SEMINAR: NEWEST TRENDS IN HIGH PERFORMANCE COMPUTING

Agenda

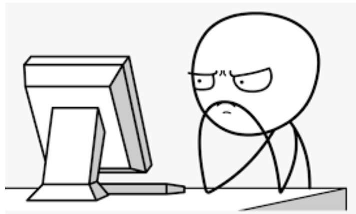
1. Introduction: The (not) perfect world of a programmer

2. OneAPI

3. Data Parallel C++

4. Practical Example: Migration of a Legacy Application

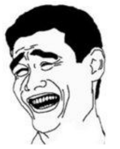
Me, during the work on this topic



Let‘s start with a question!

When developing a program, should the developer take the hardware into consideration?

Well, it depends!



Where we come from

Heavy need for prioritization

- Code useable for general hardware

  OR

- Code perfectly matching for **one** system

Worst Case: Multiple code bases for same problem!

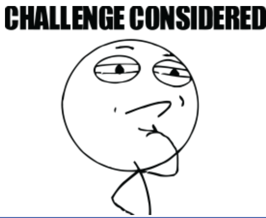
The „Holy Grail“

No matter which hardware, everything runs equally effective

- CPUs
- GPUs
- Deep-Learning-Hardware
- FPGAs (Field Programmable Gate Array)

Unique, effective communication between elements

Maybe even unique programming language?

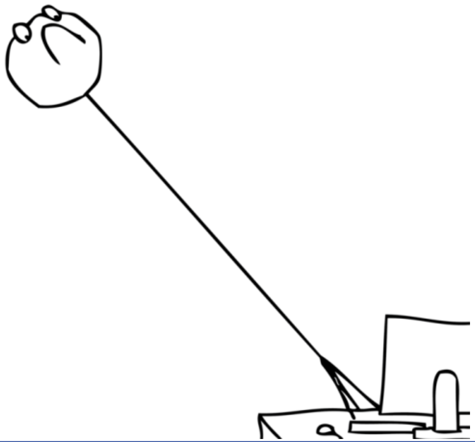


Solution: OneAPI

Developed by Intel, alpha-released 2020

“oneAPI is an open, cross-industry, standards-based, unified, multiarchitecture, multi-vendor programming model that delivers a common developer experience across accelerator architectures”

https://www.oneapi.io/



Software Model of OneAPI

Based on SYCL-Specification

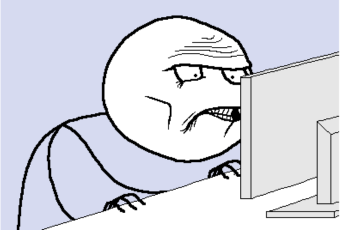
- higher-level programming model to improve programming productivity
- Cross-plattform abstraction layer

4 parts:

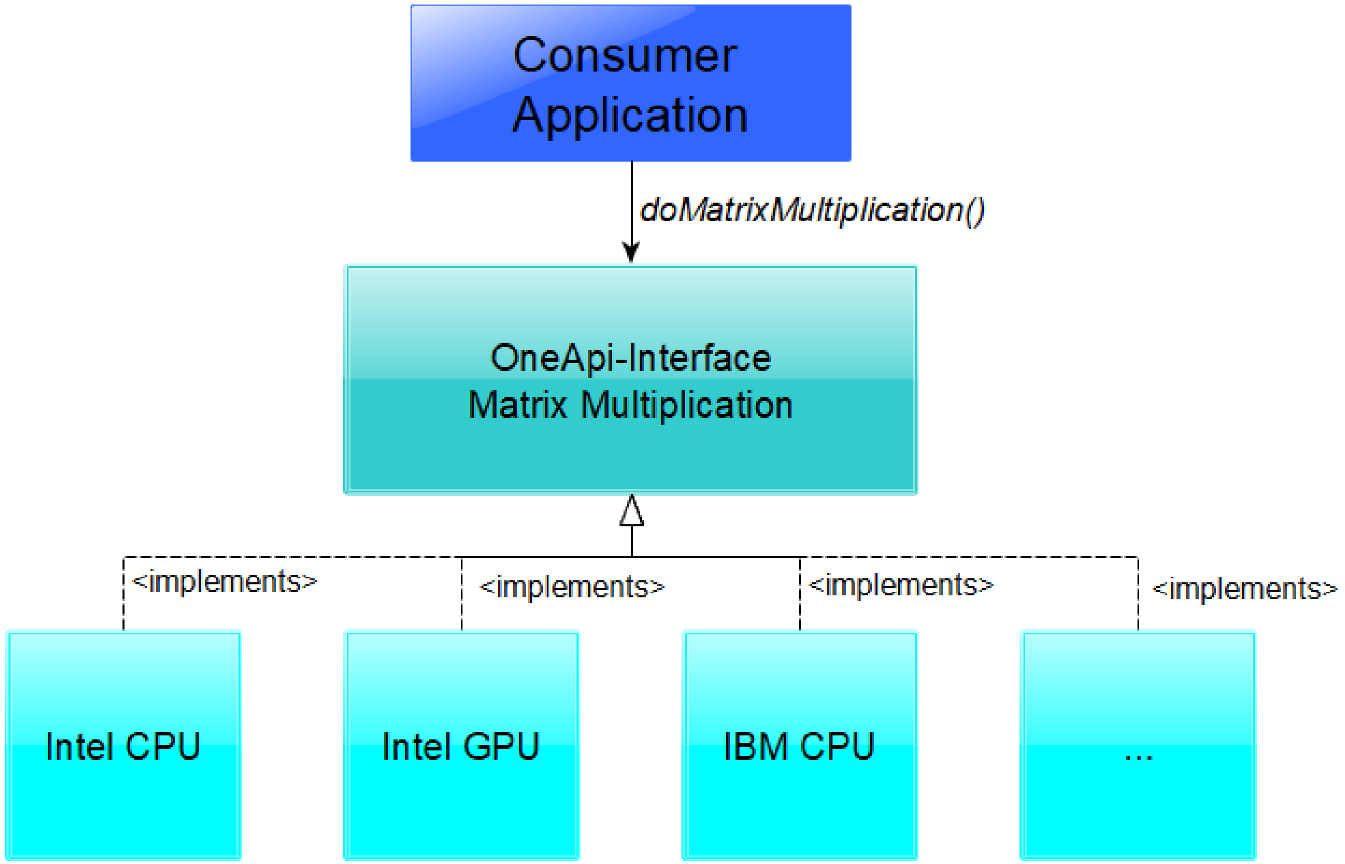
- Plattform model -> specifies host and engine
- Execution model -> holds jobs and queues them
- Storage model -> specifies storage usage
- Kernel model -> aligns kernels

Domain specific extensions

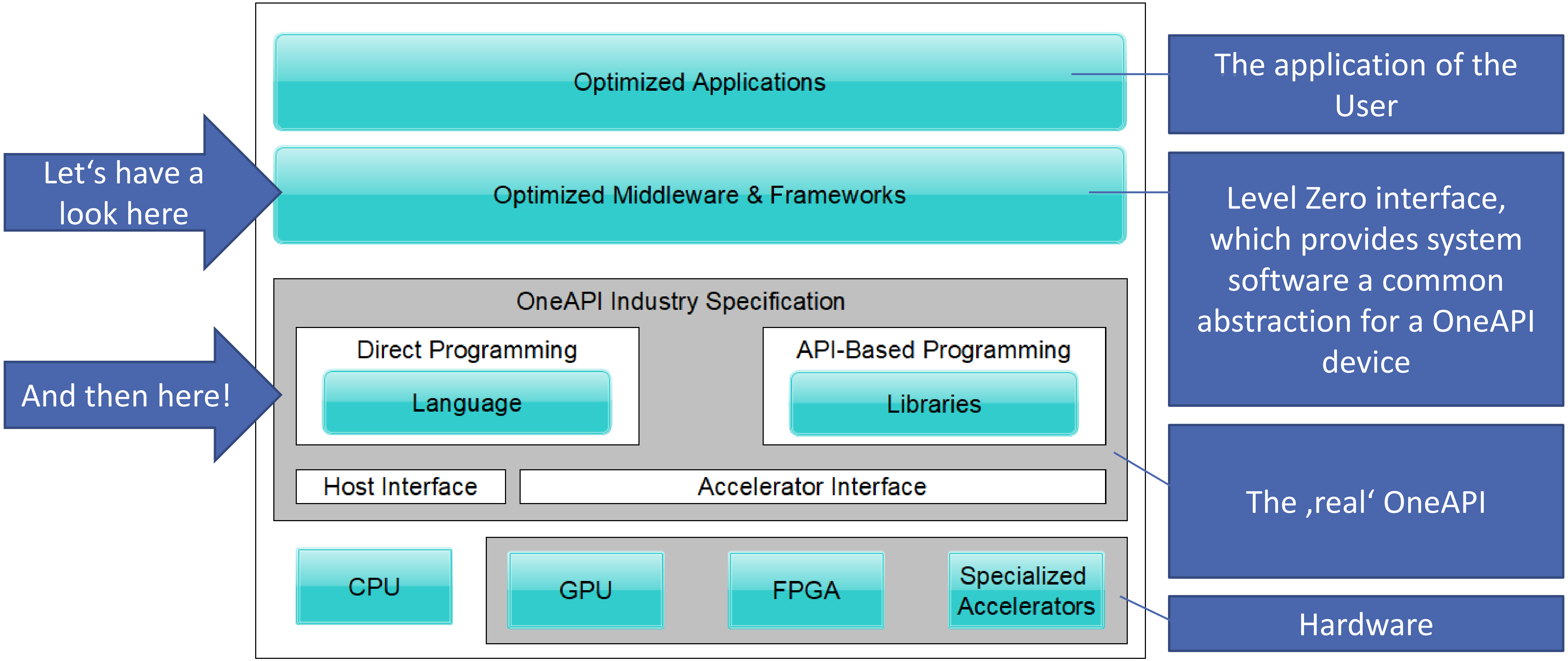
- Example: OneAPI for Video Cutting

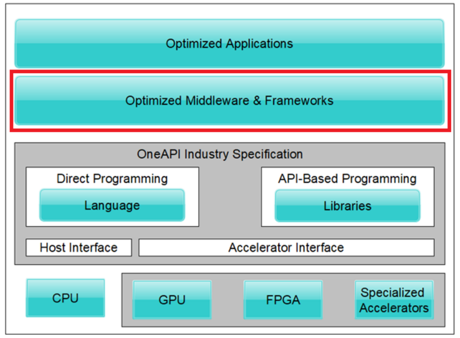


How it works


Consumer Application
doMatrixMultiplication()
OneApi-Interface
Matrix Multiplication
<implements>
<implements>
<implements>
<implements>
Intel CPU
Intel GPU
IBM CPU
...

OneAPI Architecture



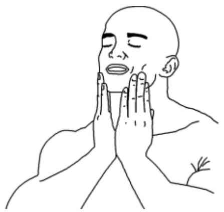


Middleware & Frameworks

Libraries using OneAPI-Interface
- Programmer can directly use them for extended functionality
- Like a python pip library, dotnet nuget-package etc.

Mainly focused on Machine Learning and Data Analytics
- oneAPI Deep Neural Network Library
- oneAPI Data Analytics Library
- oneAPI Math Kernel Library

Further libraries
- oneAPI Video Processing Library (Video editing)
- oneAPI Threading Building Blocks (Better parallel code)



Excursion: Data Parallel c++

PROGRAMMING WITH ONEAPI

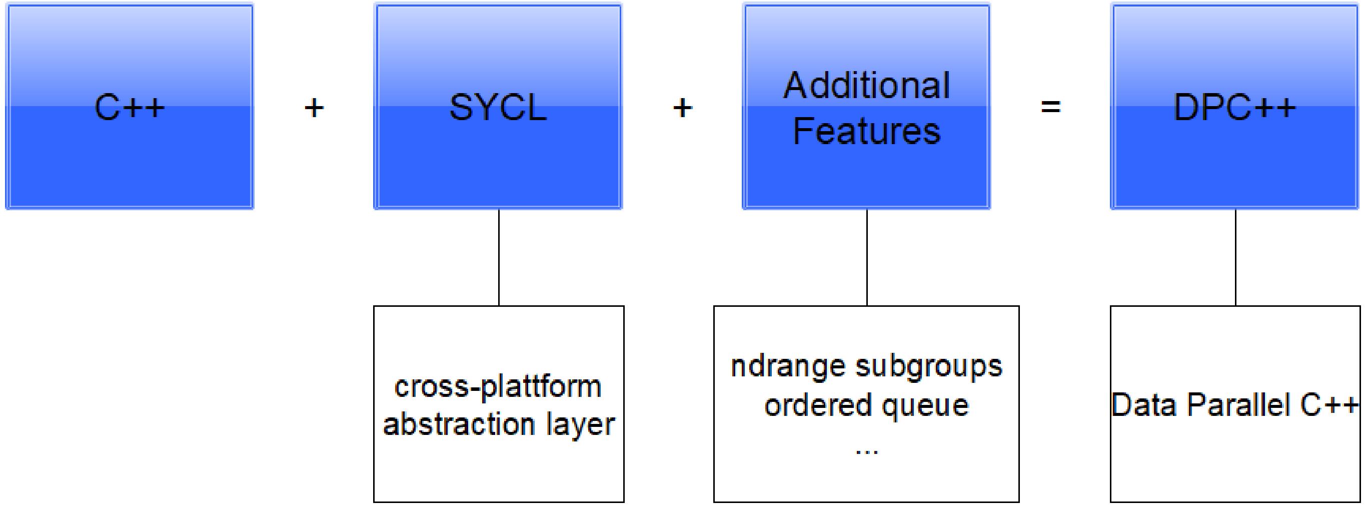
Data Parallel C++

New Cross-Architecture-Language

- Domain-specific libraries are available
- Supports development in Python

Hardware producers can implement interfaces for their components

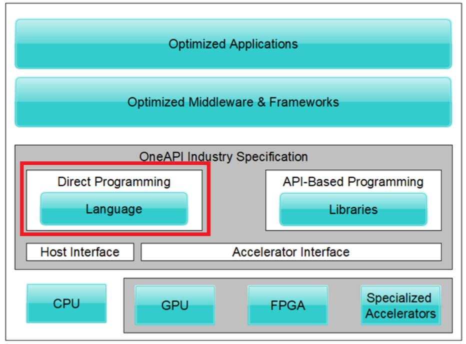
- Usage of DPC++ enables easy exchange between different parts



Data Parallel C++ - Example code

```cpp
#include <CL/sycl.hpp>
#include <iostream>

constexpr int num=16;
using namespace sycl;

int main() {
  auto r = range{num};
  buffer<int> a{r};

  queue{}.submit([&](handler& h) {
    accessor out{a, h};
    h.parallel_for(r, [=](item<1> idx) {
      out[idx] = idx;
    });
  });

  host_accessor result{a};
  for (int i=0; i<num; ++i)
    std::cout << result[i] << "\n";
}
```

What do you think is this code doing?



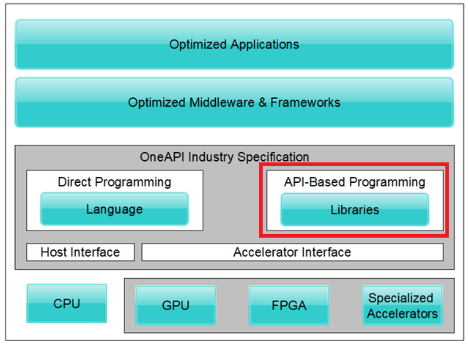
Direct Programming

Example: Simple Matrix multiplication

Use GPU & define buffers

The normal matrix multipliaction

```
using namespace cl::sycl;

// declare host arrays
double *Ahost = new double[M*N];
double *Bhost = new double[N*P];
double *Chost = new double[M*P];

{
    // Initializing the devices queue with a gpu_selector
    queue q{gpu_selector()};

    // Creating 2D buffers for matrices which are bound to host arrays
    buffer<double, 2> a{Ahost, range<2>{M,N}};
    buffer<double, 2> b{Bhost, range<2>{N,P}};
    buffer<double, 2> c{Chost, range<2>{M,P}};

    // Submitting command group to queue to compute matrix c=a*b
    q.submit([&](handler &h){
        // Read from a and b, write to c
        auto A = a.get_access<access::mode::read>(h);
        auto B = b.get_access<access::mode::read>(h);
        auto C = c.get_access<access::mode::write>(h);

        int WidthA = a.get_range()[1];

        // Executing kernel
        h.parallel_for<class MatrixMult>(range<2>{M, P}, [=](id<2> index){
            int row = index[0];
            int col = index[1];

            // Compute the result of one element in c
            double sum = 0.0;
            for (int i = 0; i < WidthA; i++) {
                sum += A[row][i] * B[i][col];
            }
            C[index] = sum;
        });
    });
}
// when we exit the block, the buffer destructor will write result back to C.
```



API-Based Programming

Example: Matrix multiplication

Use GPU and create buffers

OneAPI-Magic

```
using namespace cl::sycl;

// declare host arrays
double *A = new double[M*N];
double *B = new double[N*P];
double *C = new double[M*P];

{
    // Initializing the devices queue with a gpu_selector
    queue q{gpu_selector()};

    // Creating 1D buffers for matrices which are bound to host arrays
    buffer<double, 1> a{A, range<1>{M*N}};
    buffer<double, 1> b{B, range<1>{N*P}};
    buffer<double, 1> c{C, range<1>{M*P}};

    mkl::transpose nT = mkl::transpose::nontrans;
    // Syntax
    //   void gemm(queue &exec_queue, transpose transa, transpose transb,
    //             int64_t m, int64_t n, int64_t k, T alpha,
    //             buffer<T,1> &a, int64_t lda,
    //             buffer<T,1> &b, int64_t ldb, T beta,
    //             buffer<T,1> &c, int64_t ldc);
    // call gemm
    mkl::blas::gemm(q, nT, nT, M, P, N, 1.0, a, M, b, N, 0.0, c, M);
}
// when we exit the block, the buffer destructor will write result back to C.
```

Note: ‘onemkl’=OneAPI

Literature & Related Work

Porting a Legacy CUDA Stencil Code to oneAPI

- **Steffen Christgau, Thomas Steinke, *May 2020***
- ***Published in 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)***

Developing medical ultrasound beamforming application on GPU and FPGA using oneAPI

- Yong Wang; Yongfa Zhou, *June 2021*
- *Published in 2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*

Applying Intel's oneAPI to a machine learning case study

- Pablo Antonio Martinez, *April 2022*
- *Part of a Doctor Work, funded by the European Regional Development Fund*

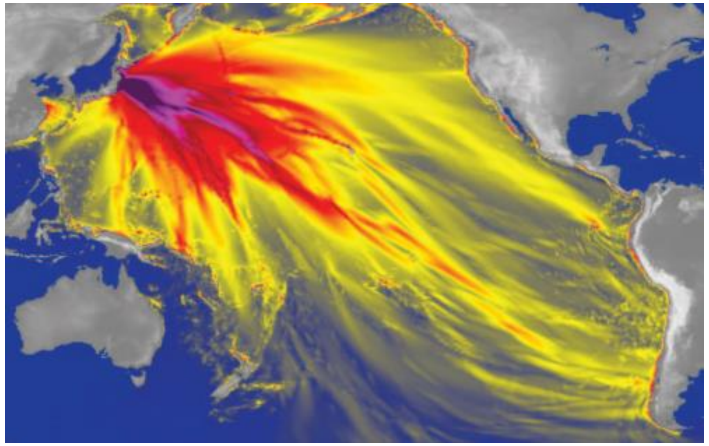
Practical Example

MIGRATION OF A LEGACY APPLICATION

About the experiment

Done at the Zuse Institute Berlin

- Supercomputer Department
- Paper: Porting a Legacy CUDA Stencil Code to oneAPI, 2020
- Authors: Steffen Christgau and Thomas Steinke



easyWave tsunami prediction

Legacy system: tsunami simulation “easyWave”

- open source
- Based on *cuda* and *OpenMP*

First decision: Rewrite code or use OneAPI Conversion Tool?

Conversion

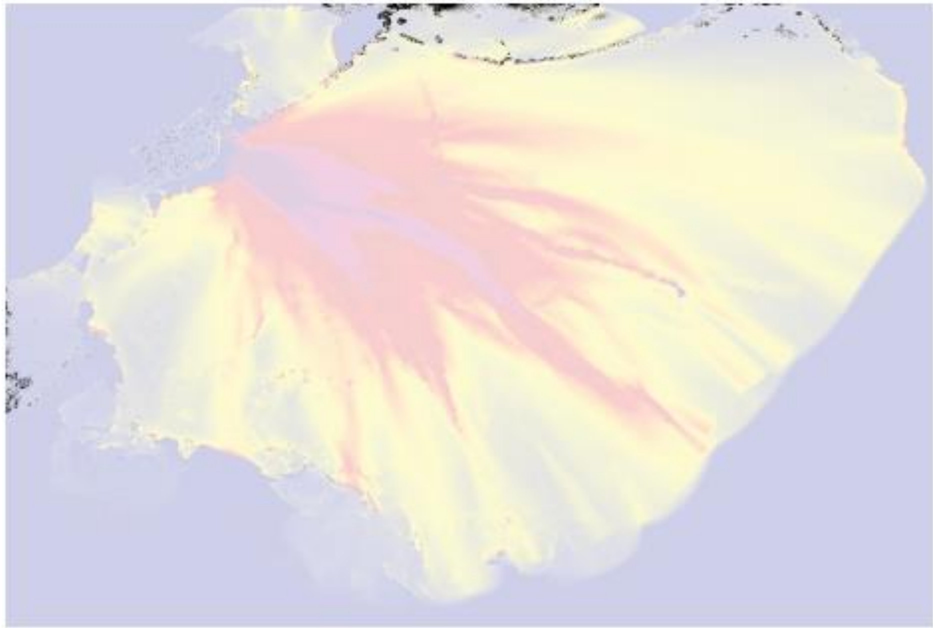
Compatibility tool only changes code related to CUDA and leaves other parts unmodified

- Very few manual modifications required
- Code still human readable
- Some minor issues, which were sent to Intel

Result was very close!

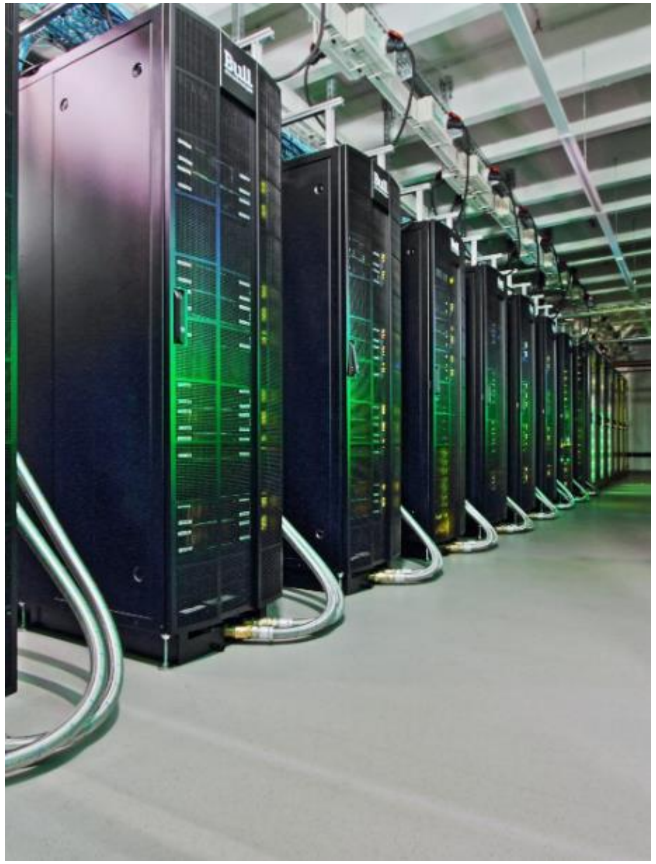
Next step: Execute on two different environments

- HLRN-IV – in Göttingen!
- OneAPI DevCloud

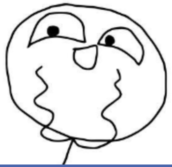


Result of converted implementation - Black dots mark differences

Plattforms used for evaluation



Parameter	HLRN-IV	DevCloud
CPU	Intel Xeon 6148 (Skylake SP)	Intel Xeon E-2176G (Coffee Lake E)
Frequency	2.4 GHz (base) 3.7 GHz (boost)	3.7 GHz (base) 4.7 GHz (boost)
Cores	2×20	1×6
Peak Perf. (SP)	1.0 TFLOP/s	259.2 GFLOP/s
Memory BW (th.)	256 GB/s	41.6 GB/s
GPU	Nvidia Tesla V100 (Volta)	Intel UHD Graphics 630 (Gen 9.5/GT2)
Frequency	1246 MHz (base) 1380 MHz (boost)	350 MHz (base) 1200 MHz (boost)
Cores	5120 FP32 Cores	24 Executions Units
Peak Perf. (SP)	14 TFLOP/s	441.6 GFLOP/s
Memory BW (th.)	900 GB/s	41.6 GB/s



OneAPI-DevCloud

"Development sandbox to learn about programming cross-architecture applications"

- Features for Basic Users:
  - Free access to Intel® oneAPI toolkits and components and the latest Intel® hardware
  - 220 GB of file storage
  - 192 GB RAM
  - 120 days of access (extensions available)
  - Terminal Interface (Linux*)
  - *Microsoft Visual Studio Code* and *Jupyter Lab* integration
    - Tutorial: https://medium.com/@kazithaque22/intel-devcloud-for-oneapi-9900ed626a8f
  - Remote Desktop for Intel® oneAPI Rendering Toolkit
- Link: https://www.intel.com/content/www/us/en/developer/tools/devcloud/overview.html

Used Versions

OpenMP SIMD

- OpenMP version with small changes for better vectorization
- Executed with different amount of threads

CUDA

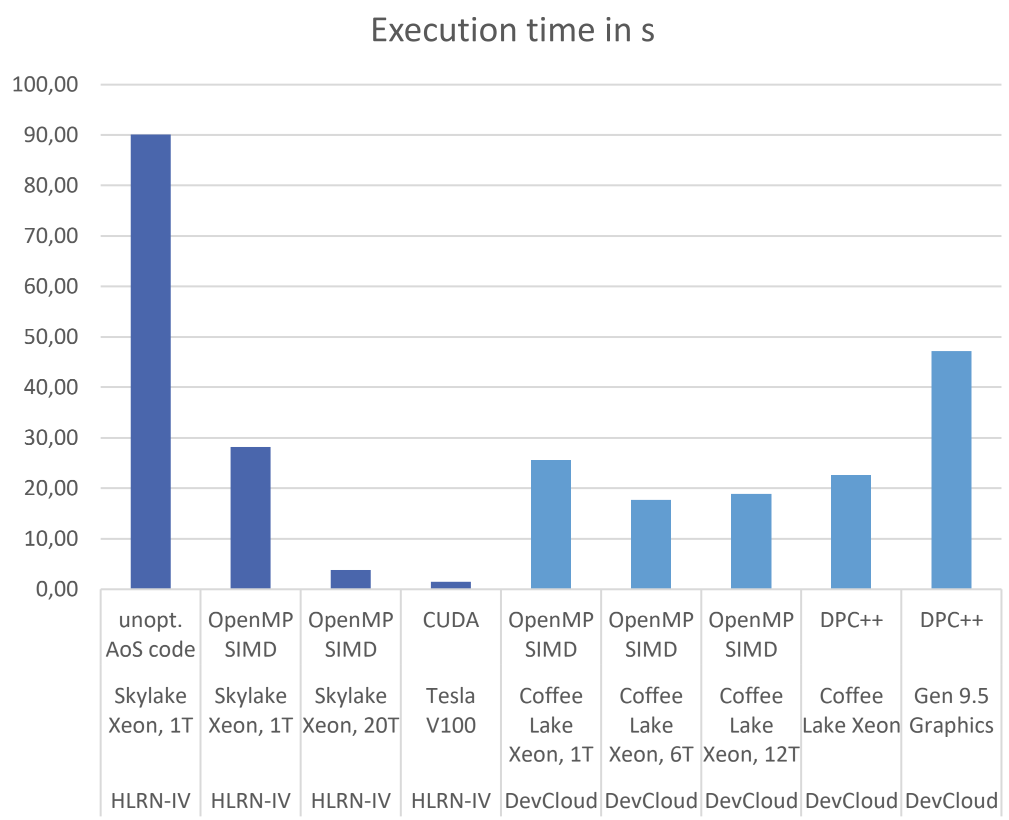
- Original, unmodified CUDA-Version

DPC++:

- Code generated by the conversion tool

Results

Environment	Hardware	Program Variant	t in s
HLRN-IV	Skylake Xeon, 1T	unopt. AoS code	90.1
HLRN-IV	Skylake Xeon, 1T	OpenMP SIMD	28.2
HLRN-IV	Skylake Xeon, 20T	OpenMP SIMD	3.8
HLRN-IV	Tesla V100	CUDA	1.5
DevCloud	Coffee Lake Xeon, 1T	OpenMP SIMD	25.6
DevCloud	Coffee Lake Xeon, 6T	OpenMP SIMD	17.7
DevCloud	Coffee Lake Xeon, 12T	OpenMP SIMD	18.9
DevCloud	Coffee Lake Xeon	DPC++	22.6
DevCloud	Gen 9.5 Graphics	DPC++	47.1



Conclusion of Experiment

Software optimised for special Hardware still better

- But: better results then the non-optimised code

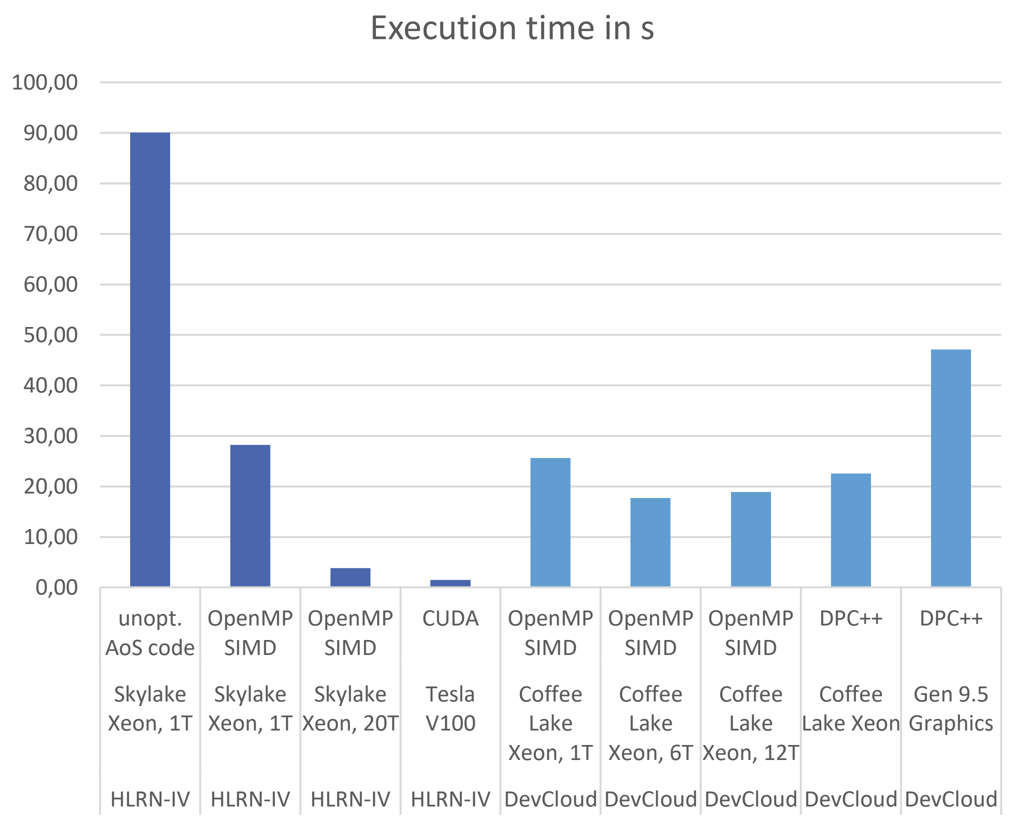
Only small performance losses even with generated code

Huge performance differences on own hardware

- DevCloud smaller differences

No optimised OneAPI Libraries were used

- Thesis: Performance will be much better with them!



Summary

OneApi: Interface for hardware-independend programming

Implementation in 3-Layers:

- Middleware & Frameworks
- OneAPI
- XPUs

Data Parallel C++

- Special C++ based language for OneAPI

Practical Experiment

- Code-Generator works!
- Results not perfect, but overall good results

OneAPI: Conclusion

The new „Holy Grail“…

… or something like… this?

Prospect

OneAPI on a good way to dominate in the future!

- Huge support from Intel
- Good test results
- Maintaining multiple hardware ressources very important

Combination with other technologies can be very promising

- Potentially huge boost for (model-driven) Infrastructure as Code

Next step for you

Do something practical and try out yourself!

- IXPUG Annual Conference 2020 – tutorials: OneAPI/ DPC++ Essential Series hands on
- https://www.youtube.com/watch?v=gCwcJNcG8Cg&ab_channel=InteleXtremePerformanceUsersGroup-IXPUG
  - Link in short: https://s.gwdg.de/mWe432
- No Hardware needed, everything on the IntelDevCloud & jupyter notebooks

Backup-Slides

2 Code Example

```
int blocks = N / 1024;

/* CUDA */
__global__ void kernel(float *v)
{
  int idx = blockIdx.x * blockSize.x + threadIdx.x;
  v[idx] = sqrt(idx * 1.0);
}

float* dev_v;
cudaMalloc(&dev_v, N * sizeof(*dev_v));
kernel<<<blocks, 1024>>>(dev_v);
cudaFree(dev_v);

/* DPC++ */
void kernel(float *v, cl::sycl::nd_item<3> item_ct1)
{
  int idx = item_ct1.get_group(0) *
    item_ct1.get_local_range().get(0) +
    item_ct1.get_local_id(0);

  v[idx] = cl::sycl::sqrt(idx * 1.0);
}

*((void **)&dev_v) = cl::sycl::malloc_device(...);
dpct::get_default_queue_wait().submit(
  [&](cl::sycl::handler &cgh) {
    auto global_rng = cl::sycl::range<3>(blocks,1,1)
      * cl::sycl::range<3>(1024, 1, 1);
    auto local_rng = cl::sycl::range<3>(1024, 1, 1);

    cgh.parallel_for<dpct_kernel_name<
        class kernel_e321fab>>(
      cl::sycl::nd_range<3>(...),
      [=](cl::sycl::nd_item<3> item_ct1) {
        kernel(dev_v, item_ct1);
      });
  }
}
cl::sycl::free(dev_v, ...);
```

3 Further environment information

Compiler Versions:

Platform	Compiler Versions
HLRN-IV	Intel Compiler 19.0.5 (for OpenMP)
HLRN-IV	GCC 7.2.0 + CUDA toolkit 10.1.105 (for GPU)
DevCloud	Intel Compiler 19.0.3 (for OpenMP)
DevCloud	oneAPI DPC++ compiler 2021.1-beta03 (2019.10.0.1106)

Maximum Relative Error compared to reference output:

Environment	Hardware	Program Variant	δ_{max}
HLRN-IV	Skylake Xeon	OpenMP SIMD	0.022
HLRN-IV	Tesla V100	CUDA	0.372
DevCloud	Coffee Lake Xeon	OpenMP SIMD	0.008
DevCloud	Coffee Lake Xeon	DPC++	0.372
DevCloud	Gen 9.5 Graphics	DPC++	0.370

Appendix

Sources

Applying Intel's oneAPI to a machine learning case study, Pablo Antonio Martínez, 2022

Porting a Legacy CUDA Stencil Code to oneAPI, Steffen Christgau; Thomas Steinke, 2020

https://www.oneapi.io/

https://www.heise.de/news/Intel-veroeffentlicht-oneAPI-Spezifikation-1-0-4914674.html

https://www.sigs-datacom.de/trendletter/2021-20/4-was-ist-oneapi

Image Sources

Intel Logo:

- https://upload.wikimedia.org/wikipedia/commons/thumb/0/0e/Intel_logo_%282020%2C_light_blue%29.svg/1200px-Intel_logo_%282020%2C_light_blue%29.svg.png

GWDG-Cluster

- https://www.gwdg.de/de_DE/hpc

OneAPI

- https://www.oneapi.io/

Tsunami Predictions:

- Porting a Legacy CUDA Stencil Code to oneAPI, Steffen Christgau; Thomas Steinke, 2020

Memes & Rage-Faces:

- https://imgflip.com/memegenerator/

Youtube-Thumbnail

- https://www.youtube.com/watch?v=gCwcJNcG8Cg&ab_channel=InteleXtremePerformanceUsersGroup-IXPUG