oneAPI

Molecular Dynamics using SYCL and complete offload analysis

With kernel fusion

Abhishek Nandy
Chief Data Scientist PrediQt Business Solution pvt ltd

What is Molecular Dynamics

- Molecular Dynamics is a computer simulation method for studying the physical movements of atoms and molecules.
- It allows us to observe the time-dependent behavior of a molecular system, providing information at atomic-level resolution.

Components of Molecular Dynamics Simulations

- System Setup: Define the initial state of all atoms in the system (positions, velocities).
- Force Calculation: Calculate forces acting on each atom using potential energy functions.
- Time Integration: Update velocities and positions of atoms using the calculated forces.
- Analysis: Examine the resulting trajectories to calculate macroscopic properties.

Practical uses of Molecular Dynamics

- Drug Discovery and Design :In the pharmaceutical industry, molecular dynamics simulations are extensively used to study the behavior of biological macromolecules and their interactions with potential drug molecules.
- Lead Optimization: During drug discovery, once a lead compound has been identified (i.e., a molecule that has some therapeutic effect), molecular dynamics can be used to optimize this lead. The MD simulation can provide insights into how minor alterations in the molecular structure of the lead compound can impact its binding affinity with the target protein. These simulations help in designing more potent and selective drugs.
- In addition to drug discovery, molecular dynamics simulations also find applications in many other fields such as materials science, enzyme engineering, and studying climate change at the molecular level. The possibilities are virtually limitless, given the broad applicability of these simulations in understanding molecular interactions and dynamics.

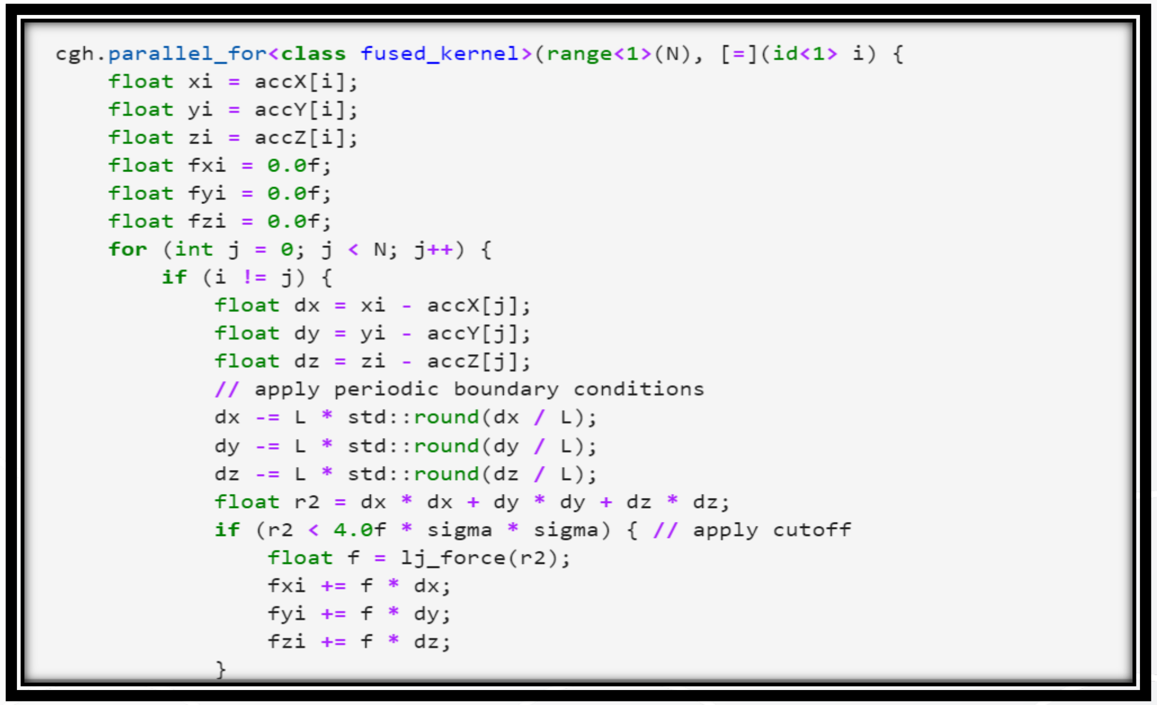
Why use SYCL in Molecular Dynamics

- Performance Portability: SYCL's heterogeneous model allows programs to be written once and then run efficiently across a range of hardware, including CPUs, GPUs, and FPGAs. This is crucial in MD simulations, which can be computationally intensive and require significant resources.
- Flexibility: SYCL's single-source development model allows for easy integration with existing C++ libraries and frameworks, simplifying the programming process and accelerating development.
- Scalability: With SYCL, it's easier to scale computations across multiple devices, which can be a significant advantage for large-scale MD simulations

Kernel fusion implementation

- Performance Improvement: Kernel fusion can significantly reduce memory latency and increase the efficiency of MD simulations, which are computationally intensive and require frequent interactions among atoms (kernels).
- Optimized Memory Usage: By reducing data transfer between the device and the host, kernel fusion can lead to better memory bandwidth utilization and enhanced simulation speed.
- Reduced Overhead: Fusion of kernels reduces the overhead associated with launching multiple kernels, leading to more efficient computations.
- SYCL's flexible, single-source C++ framework allows for easy integration of kernel fusion techniques, providing developers with a powerful tool for optimizing performance in MD simulations.

Code explanation and demo



```
cgh.parallel_for<class fused_kernel>(range<1>(N), [=](id<1> i) {
    float xi = accX[i];
    float yi = accY[i];
    float zi = accZ[i];
    float fxi = 0.0f;
    float fyi = 0.0f;
    float fzi = 0.0f;
    for (int j = 0; j < N; j++) {
        if (i != j) {
            float dx = xi - accX[j];
            float dy = yi - accY[j];
            float dz = zi - accZ[j];
            // apply periodic boundary conditions
            dx -= L * std::round(dx / L);
            dy -= L * std::round(dy / L);
            dz -= L * std::round(dz / L);
            float r2 = dx * dx + dy * dy + dz * dz;
            if (r2 < 4.0f * sigma * sigma) { // apply cutoff
                float f = lj_force(r2);
                fxi += f * dx;
                fyi += f * dy;
                fzi += f * dz;
            }
```

In Context: The Code Snippet

- Demonstrates an implementation of a Molecular Dynamics (MD) simulation using SYCL with kernel fusion.
- The loop within cgh.parallel_for<class fused_kernel> block performs several tasks:
- Force Calculation: Computes the force on each particle due to every other particle.
- Velocity & Position Update: Adjusts the velocity based on the force, and updates the particle's position.
- Applying Periodic Boundary Conditions: Ensures particles stay within the simulation box.

Continued

Benefits in This Case

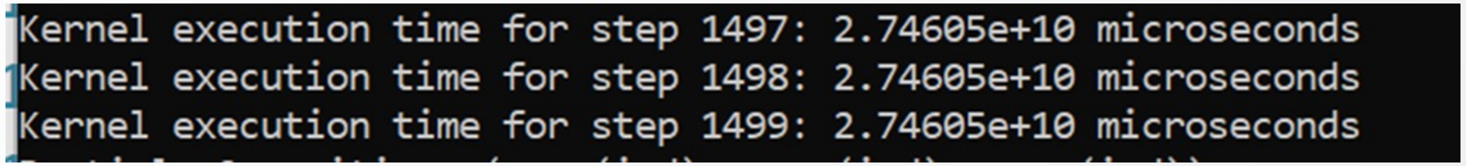
Combining these tasks into a single fused kernel improves efficiency by reducing the overhead of launching separate kernels and lessens memory transfers between host and device.

- Performance Metrics

After kernel execution, the code calculates and prints execution time, showcasing performance improvements through kernel fusion.

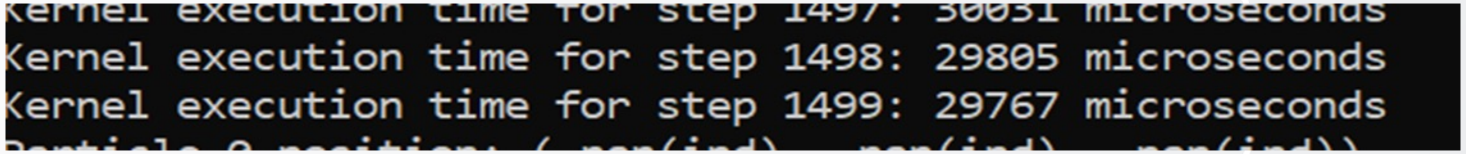
Performance matrix with and without Fused kernel

- Without fused kernel.



```
Kernel execution time for step 1497: 2.74605e+10 microseconds
Kernel execution time for step 1498: 2.74605e+10 microseconds
Kernel execution time for step 1499: 2.74605e+10 microseconds
```

- With fused kernel



```
Kernel execution time for step 1497: 30031 microseconds
Kernel execution time for step 1498: 29805 microseconds
Kernel execution time for step 1499: 29767 microseconds
```

Offloading to Intel Advisor

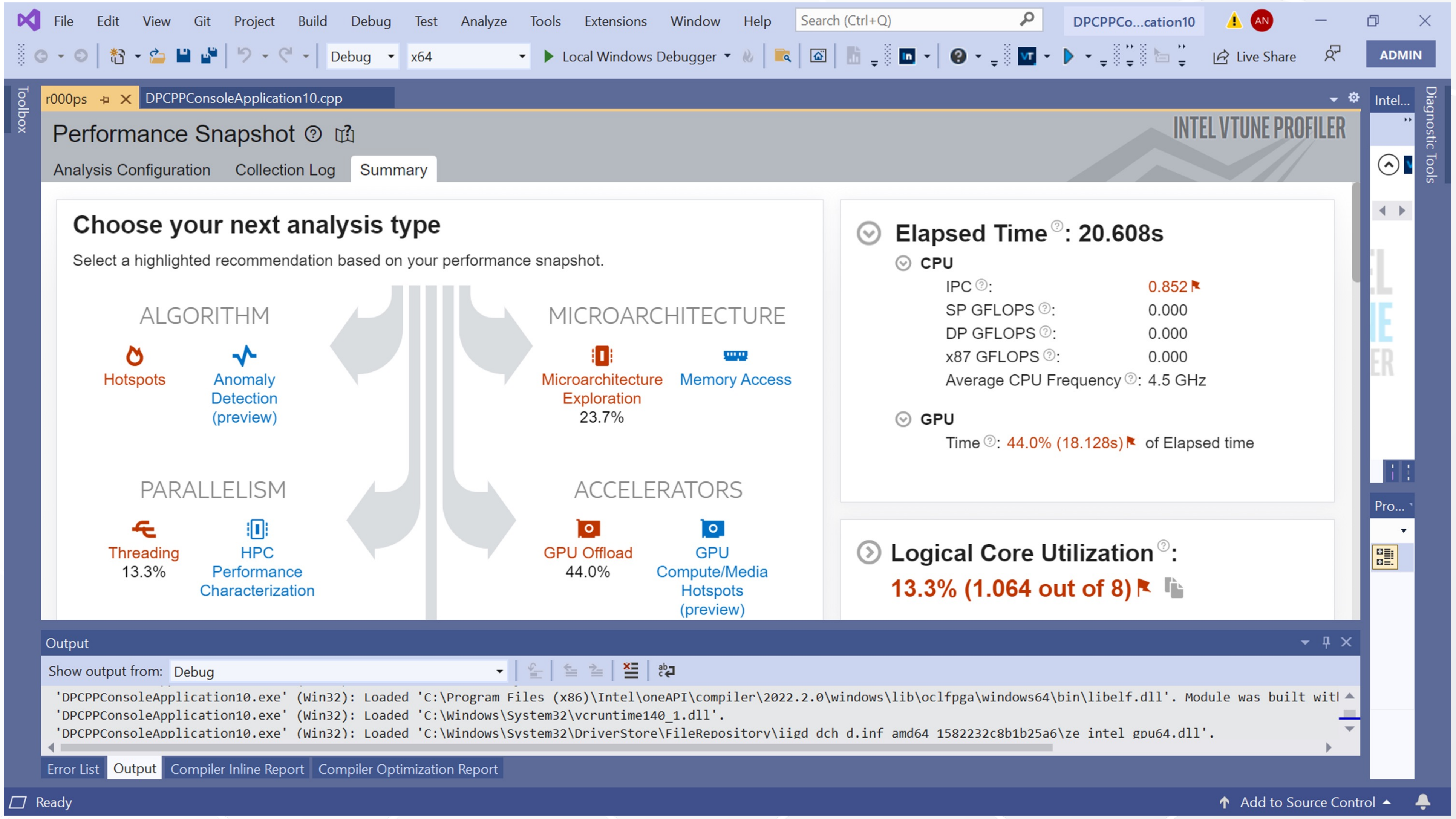
It achieves this through features like:

Performance Profiling: It enables you to profile the performance of your SYCL kernels and spot bottlenecks or inefficiencies in your code.

Offload Modeling: It can simulate what performance might look like if you offloaded certain parts of your code to different hardware, such as CPUs, GPUs, or FPGAs. This can help you make decisions about how to partition your code for heterogeneous computing.

Vectorization Optimization: It helps analyze and optimize data parallelism in your code to make effective use of vectorized instructions on Intel hardware, thus improving execution speed.

These features and insights provided by Intel Advisor can guide you to refactor or tune your SYCL code to achieve better performance on your target Intel hardware.



Performance Snapshot

Analysis Configuration | Collection Log | Summary

Choose your next analysis type

Select a highlighted recommendation based on your performance snapshot.

ALGORITHM
- Hotspots
- Anomaly Detection (preview)

MICROARCHITECTURE
- Microarchitecture Exploration 23.7%
- Memory Access

PARALLELISM
- Threading 13.3%
- HPC Performance Characterization

ACCELERATORS
- GPU Offload 44.0%
- GPU Compute/Media Hotspots (preview)

Elapsed Time: 20.608s

CPU

IPC:	0.852
SP GFLOPS:	0.000
DP GFLOPS:	0.000
x87 GFLOPS:	0.000
Average CPU Frequency:	4.5 GHz

GPU

Time: 44.0% (18.128s) of Elapsed time

Logical Core Utilization:

13.3% (1.064 out of 8)

Output

Show output from: Debug

```
'DPCPPConsoleApplication10.exe' (Win32): Loaded 'C:\Program Files (x86)\Intel\oneAPI\compiler\2022.2.0\windows\lib\oclfpga\windows64\bin\libelf.dll'. Module was built wit
'DPCPPConsoleApplication10.exe' (Win32): Loaded 'C:\Windows\System32\vcruntime140_1.dll'.
'DPCPPConsoleApplication10.exe' (Win32): Loaded 'C:\Windows\System32\DriverStore\FileRepository\iigd_dch_d.inf_amd64_1582232c8b1b25a6\ze_intel_gpu64.dll'.
```

Results and summarization

Kernel fusion helps in performance by reducing the overhead of launching multiple kernels and minimizing memory traffic between the CPU and GPU, as fused operations can share data locally without frequent communication.

For this specific molecular dynamics simulation, Intel Advisor can identify computational hotspots and offer advice on how to improve parallelism. It can suggest optimization techniques to better use the available hardware, such as vectorization, offloading parts of the computations to an accelerator like a GPU or an FPGA, and optimization of memory access patterns.

Intel Advisor can also provide a Roofline model analysis, helping you understand where your code stands in terms of performance ceilings and what you could theoretically achieve with perfect optimization.

Coordinates and code

Code

https://github.com/AbhiLegend/MolecularDynamicsSYCLConference

Linkedin

https://in.linkedin.com/in/abhishek-nandy?original_referer=https%3A%2F%2Fwww.google.com%2F

Youtube channel link

https://www.youtube.com/channel/UCD1IBC7I6QNpPNMYjubi0tg

Call to action

- Deep Dive into SYCL: Expand your knowledge and understanding of SYCL. Harness its power to improve your applications.
- https://github.com/intel/llvm/tree/sycl
- Revisit the Topics Covered: Review the topics we've discussed during this summit. There's always more to learn and understand!
- Join Our Community: Be a part of our ever-growing community. Collaborate, learn, and grow with us.
- https://devmesh.intel.com/
- Stay Updated: Keep an eye out for our upcoming events, webinars, and more.

Thank you