Experiences tuning SYCL libraries

John Lawson

P3HPC Forum – Sept 2020
Auto-tuning compute kernels

• Parameterized kernels allow performance to be tuned for a range of hardware
• Automatically tuning these parameters reduces developer effort and increases effectiveness
• Auto-tuners work out the best set of kernel parameters for a given set of inputs...
• ... but general purpose libraries want to provide performance on all possible input sizes
Auto-tuning OpenCL

• OpenCL kernels are provided as source code, with parameters set using the preprocessor
• Cost of using different kernel parameters is only JIT compilation time
Required steps

1. Use auto-tuning to find kernel parameters for a representative range of input sizes
2. Provide a system to choose optimal kernel parameters for unseen input sizes
Existing implementations

• CLBlast
  • Provides a database of devices and tuning scripts
  • Uses a single best configuration for each device

• clBlas
  • Provides a number of different kernels generated for library targets

• ARM Compute Library
  • Hardcodes kernels and kernel parameters for the library targets
• SYCL is a single-source heterogeneous parallel programming model maintained by the Khronos Group
• Allows developers to write compute kernels in C++
Providing kernels in a SYCL library

• SYCL kernels compiled to bitcode (SPIR, SPIR-V, PTX, GCN,...)
• Each tuned kernel a binary blob embedded in the library
• More kernels = better performance, but also larger binaries
SYCL required steps

1. Use auto-tuning to find kernel parameters for representative input sizes
2. Choose a subset of kernel parameters to deploy in library
3. Create a system to choose optimal kernel parameters for unseen inputs
Matrix multiply case study

template<typename T, typename Index, bool TransposeLHS, bool TransposeRHS,
    int RowTile, int AccTile, int ColTile>
struct MatmulKernel;

• 640 possible kernel configurations
  • Tile sizes 1, 2, 4 and 8
  • Work-group sizes of 1, 8, 16, 32 and 128

• Recorded average execution time and flops for 300 matrix sizes on AMD R9 Nano GPU and Intel i7-6700K CPU.
Frequency that one kernel is best

GPU:

CPU:
Selecting a subset of kernels (GPU)
Selecting a subset of kernels (CPU)
Selecting kernels at runtime

Percent of optimal performance

GPU:

CPU:

Number of kernel configurations
Performance on machine learning model

Time for one image inference of a SYCL-DNN implementation of VGG16 with different matrix multiplication libraries.
Conclusions

• Unsupervised machine learning techniques provide easy and effective methods to select a subset of kernels to deploy in a SYCL library.

• A decision tree gives reasonable performance when selecting which of these kernels to use at runtime.

• This system extracts portable performance from parameterized SYCL kernels, beating other tuned libraries.