Collabora Productivity

The Future of OpenCL in LibreOffice

Tor Lillqvist

Collabora Productivity

@TorLillqvist

What is OpenCL

- Vendor-neutral, architecture-agnostic programming language for parallelized computation
- Available on the three desktop platforms we care about
- In practice, relevant mostly on Windows
  - macOS: Apple not really that keen any longer
  - Linux: a mess

What is OpenCL

- OpenCL code is quite C-like, with some extra keywords and library functions related to numeric calculations and for moving data in/out of kernels
- The unit of execution is called a “kernel”
- A kernel is roughly equivalent to an OpenGL shader
- Kernels compiled at run-time. Can also be saved as platform-dependent binaries and loaded from such

What does OpenCL look like?

Sample FFT kernel

```
// This kernel computes FFT of length 1024. The 1024 length FFT is decomposed into
// calls to a radix 16 function, another radix 16 function and then a radix 4 function

__kernel void fft1D_1024 (__global float2 *in, __global float2 *out,
                          __local float *sMemx, __local float *sMemy) {
  int tid = get_local_id(0);
  int blockIdx = get_group_id(0) * 1024 + tid;
  float2 data[16];

  // starting index of data to/from global memory
  in = in + blockIdx;  out = out + blockIdx;

  globalLoads(data, in, 64); // coalesced global reads
  fftRadix16Pass(data);      // in-place radix-16 pass
  twiddleFactorMul(data, tid, 1024, 0);

  // local shuffle using local memory
  localShuffle(data, sMemx, sMemy, tid, (((tid & 15) * 65) + (tid >> 4)));
  fftRadix16Pass(data);               // in-place radix-16 pass
  twiddleFactorMul(data, tid, 64, 4); // twiddle factor multiplication

  localShuffle(data, sMemx, sMemy, tid, (((tid >> 4) * 64) + (tid & 15)));

  // four radix-4 function calls
  fftRadix4Pass(data);      // radix-4 function number 1
  fftRadix4Pass(data + 4);  // radix-4 function number 2
  fftRadix4Pass(data + 8);  // radix-4 function number 3
  fftRadix4Pass(data + 12); // radix-4 function number 4
```

What does OpenCL look like?

Host code to call it

```
// create a compute context with GPU device
context = clCreateContextFromType(NULL, CL_DEVICE_TYPE_GPU, NULL, NULL, NULL);

// create a command queue
clGetDeviceIDs( NULL, CL_DEVICE_TYPE_DEFAULT, 1, &device_id, NULL );
queue = clCreateCommandQueue(context, device_id, 0, NULL);

// allocate the buffer memory objects
memobjs[0] = clCreateBuffer(context, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
sizeof(float)*2*num_entries, srcA, NULL);
memobjs[1] = clCreateBuffer(context, CL_MEM_READ_WRITE, sizeof(float)*2*num_entries, NULL, NULL);

// create the compute program
program = clCreateProgramWithSource(context, 1, &fft1D_1024_kernel_src, NULL, NULL);

// build the compute program executable
clBuildProgram(program, 0, NULL, NULL, NULL, NULL);

// create the compute kernel
kernel = clCreateKernel(program, "fft1D_1024", NULL);

// set the args values
clSetKernelArg(kernel, 0, sizeof(cl_mem), (void *)&memobjs[0]);
clSetKernelArg(kernel, 1, sizeof(cl_mem), (void *)&memobjs[1]);
clSetKernelArg(kernel, 2, sizeof(float)*(local_work_size[0]+1)*16, NULL);
clSetKernelArg(kernel, 3, sizeof(float)*(local_work_size[0]+1)*16, NULL);

// create N-D range object with work-item dimensions and execute kernel
global_work_size[0] = num_entries;
local_work_size[0] = 64; //Nvidia: 192 or 256
clEnqueueNDRangeKernel(queue, kernel, 1, NULL, global_work_size, local_work_size, 0, NULL, NULL);
```

What does OpenCL look like?

- OpenCL in LibreOffice uses kernels generated at run-time, i.e. compiled from Calc formulas
- Both the OpenCL-generating code and the resulting OpenCL source are quite complex
- Debugging of OpenCL means modifying the C++ code to emit printf() calls in generated OpenCL

What does OpenCL look like?

LibreOffice OpenCL generating code

```
void OpAverageA::GenSlidingWindowFunction(
    std::stringstream &ss, const std::string &sSymName, SubArguments &vSubArguments)
{
    int isMixed = 0;
    ss << "\ndouble " << sSymName;
    ss << "_"<< BinFuncName() <<"(";
    for (size_t i = 0; i < vSubArguments.size(); i++)
    {
        if (i)
            ss << ",";
        vSubArguments[i]->GenSlidingWindowDecl(ss);
    }
    ss << ")\n";
    ss << "{\n";
    ss << "    int gid0=get_global_id(0);\n";
    ss << "    double tmp0 = 0.0;\n";
    ss << "    double nCount = 0.0;\n";
    ss <<"\n";
...
            ss << "    for (int i = ";
            if (!pDVR->IsStartFixed() && pDVR->IsEndFixed()) {
                ss << "gid0; i < " << pDVR->GetArrayLength();
                ss << " && i < " << nCurWindowSize  << "; i++){\n";
            } else if (pDVR->IsStartFixed() && !pDVR->IsEndFixed()) {
                ss << "0; i < " << pDVR->GetArrayLength();
                ss << " && i < gid0+"<< nCurWindowSize << "; i++){\n";
```

What does OpenCL look like?

Generated OpenCL code

```
double tmp0_0_average(__global double *tmp0_0_0) {
double tmp = 0;
int gid0 = get_global_id(0);
int nCount = 0;
double tmpBottom;
tmpBottom = 0;
        {int i;
        i = 0;
        if(i + gid0 < 220){
                tmp = legalize(fsum_count(tmp0_0_0[i + gid0],tmp, &nCount), tmp);
                        }
        i = 1;
        if(i + gid0 < 220){
                tmp = legalize(fsum_count(tmp0_0_0[i + gid0],tmp, &nCount), tmp);
                        }
        i = 2;
        if(i + gid0 < 220){
                tmp = legalize(fsum_count(tmp0_0_0[i + gid0],tmp, &nCount), tmp);
                        }
        i = 3;
        if(i + gid0 < 220){
                tmp = legalize(fsum_count(tmp0_0_0[i + gid0],tmp, &nCount), tmp);
                        }
...
        i = 9;
        if(i + gid0 < 220){
                tmp = legalize(fsum_count(tmp0_0_0[i + gid0],tmp, &nCount), tmp);
                        }
        }
if (nCount==0)
   return CreateDoubleError(errDivisionByZero);
return tmp*pow((double)nCount,-1.0);
}
```

The Past

- Formula Group: new concept in Calc and its import filters
- When several contiguous cells in a column are effectively the same formula, a single “formula group” is used
- (Formula groups also used by the so-called software interpreter, which does not use OpenCL, but SIMD instructions, for long SUM() formulas mainly)

The Past

- OpenCL implementation of most Calc operators and functions (“opcodes”)
- Even formulas using strings were thought to be suitable for OpenCL. Strings were UPPER-CASED (!)
- Incomplete unit tests. Corner cases not checked: Strings to be interpreted as numbers, empty cells, empty string handling modes, error cases like #DIV/0!
- Problems all over the place

The Past

- First attempt at sanity: Use OpenCL only for formulas that use only “simple” opcodes that can be checked for correctness, and only for formula groups that are larger than a minimum size
- Make the subset of opcodes and the minimum size user-visible options
- Whitelist and blacklist of OpenCL vendor implementations, also user-visible options

The Current

- Many corner case bugs fixed
- We no longer try to use OpenCL for strings
- Those OpenCL implementations that are “trusted” have now been fairly well tested

The Current

- The user-visible options dropped. There is no reason to let users try to use untested likely broken code that might silently corrupt their data
- OpenCL platform (driver) correctness is tested at first start of fresh installation (or profile). If found to be problematic, OpenCL usage turned silently off

The Future

- OpenCL will be continued to be used for well-tested Calc formula opcodes
- OpenCL could be used also for other calculations where parallelisation could help performance significantly
  - Image format en/decoding?
  - But: Most core developers have no useful OpenCL access, and OpenCL has bad “reputation” among them already

Collabora

- Collabora Ltd.
  - Leading Open Source Consultancy
  - 10 years of experience. 90+ People.

- Collabora Productivity Ltd.
  - Dedicated to Enterprise LibreOffice
  - Provides Level-3 support (code issues) to all SUSE LibreOffice clients
  - Architects of Microsoft OpenXML filters

FIN