Computational Science on Many-Core Architectures

360.252

Karl Rupp

Institute for Microelectronics
Vienna University of Technology
http://www.iue.tuwien.ac.at

Zoom Channel 95028746244
Wednesday, December 16, 2020

Agenda for Today

Exercise 8 Recap

OpenMP for GPUs

Profiling and Debugging

Exercise 9

Exercise 8 Recap

Kernel

- How was your experience?

Exercise 8 Recap

Kernel

- How was your experience?
- Points for Exercise 7 will be provided within 24 hours.

Exercise 8 Recap

Kernel

- How was your experience?
- Points for Exercise 7 will be provided within 24 hours.

Reference Solution: ViennaCL

```
viennacl::vector<double> x(N);
viennacl::vector<double> y(N);
...
double dot = viennacl::linalg::inner_prod(x + y, x - y);
```

Reference Solution: VexCL

```
vex::vector<double> dev_x(ctx, x);
vex::vector<double> dev_y(ctx, y);
vex::Reductor<double, vex::SUM> sum(ctx);
double dot = sum((dev_x + dev_y) * (dev_x - dev_y));
```

Exercise 8 Recap

Reference Solution: Thrust

```
  thrust::transform(thrust::device, x.begin(), x.end(), y.begin(),
                    tmp_1.begin(), thrust::plus<double>());
  thrust::transform(thrust::device, x.begin(), x.end(), y.begin(),
                    tmp_2.begin(), thrust::minus<double>());
  dot = thrust::inner_product(thrust::device, tmp_1.begin(),
                    tmp_1.end(), tmp_2.begin(), 0);
```

Reference Solution: Boost.Compute

```
boost::compute::transform(x.begin(), x.end(), y.begin(),
                  tmp1.begin(), boost::compute::plus<double>(),
                  queue);
boost::compute::transform(x.begin(), x.end(), y.begin(),
                  tmp2.begin(), boost::compute::minus<double>(),
                  queue);
dot = boost::compute::inner_product(tmp1.begin(), tmp1.end(),
                                    tmp2.begin(), 0, queue);
```

Apparent Problem: Temporary vectors!

Exercise 8 Recap

Math Tweak

- Looking at the math can yield interesting shortcuts
- Tweak by David Fischak (and possibly others):

$$\langle x + y, x - y \rangle = \|x\|^2 - \|y\|^2$$

OpenMP for GPUs

About

- Compiler directives (code annotations)
- Offload `target` introduced in OpenMP 4.5
- Example (still hard to find):

```
double *data_array = (double*)malloc(sizeof(double) * N);
for (size_t i=0; i<N; ++i) data_array[i] = i;

double red = 0;
#pragma omp target teams distribute parallel for map(to:
    data_array[0:N]) map(tofrom: red) reduction(+:red)
for (int idx = 0; idx < N; ++idx)
{
    red += data_array[idx];
}
```

OpenMP for GPUs

About

- Compiler directives (code annotations)
- Offload `target` introduced in OpenMP 4.5
- Example (still hard to find):

```
double *data_array = (double*)malloc(sizeof(double) * N);
for (size_t i=0; i<N; ++i) data_array[i] = i;

double red = 0;
#pragma omp target teams distribute parallel for map(to:
    data_array[0:N]) map(tofrom: red) reduction(+:red)
for (int idx = 0; idx < N; ++idx)
{
    red += data_array[idx];
}
```

- Noteworthy details:
  1. `omp target`: Run on the GPUs
  2. `teams distribute`: Spread threads over the GPU
  3. `map()`: Move data to or from GPU

OpenMP for GPUs

Let’s write an OpenMP-parallel library!

Threads and Library Interfaces

Attempt 1

- Library spawns threads

```
void library_func(double *x, int N) {
  #pragma omp parallel for
  for (int i=0; i<N; ++i) x[i] = something_complicated();
}
```

Threads and Library Interfaces

Attempt 1

- Library spawns threads

```
void library_func(double *x, int N) {
  #pragma omp parallel for
  for (int i=0; i<N; ++i) x[i] = something_complicated();
}
```

Problems

- Call from multi-threaded environment?

```
void user_func(double **y, int N) {
  #pragma omp parallel for
  for (int j=0; j<M; ++j) library_func(y[j], N);
}
```

- Incompatible OpenMP runtimes (e.g. GCC vs. ICC)

Threads and Library Interfaces

Attempt 2

- Use pthreads/TBB/etc. instead of OpenMP to spawn threads
- Fixes incompatible OpenMP implementations (probably)

Threads and Library Interfaces

Attempt 2

- Use pthreads/TBB/etc. instead of OpenMP to spawn threads
- Fixes incompatible OpenMP implementations (probably)

Problems

- Still a problem with multi-threaded user environments

```
void user_func(double **y, int N) {
  #pragma omp parallel for
  for (int j=0; j<M; ++j) library_func(y[j], N);
}
```

Threads and Library Interfaces

Attempt 3

- Hand back thread management to user

```
void library_func(ThreadInfo ti, double *x, int N) {
  int start = compute_start_index(ti, N);
  int stop  = compute_stop_index(ti, N);
  for (int i=start; i<stop; ++i)
    x[i] = something_complicated();
}
```

Threads and Library Interfaces

Attempt 3

- Hand back thread management to user

```
void library_func(ThreadInfo ti, double *x, int N) {
  int start = compute_start_index(ti, N);
  int stop  = compute_stop_index(ti, N);
  for (int i=start; i<stop; ++i)
    x[i] = something_complicated();
}
```

Implications

- Users can use their favorite threading model
- API requires one extra parameter
- Extra boilerplate code required in user code

Threads and Library Interfaces

Reflection

- Extra thread communication parameter

```
void library_func(ThreadInfo ti, double *x, int N) {...}
```

Threads and Library Interfaces

Reflection

- Extra thread communication parameter

```
void library_func(ThreadInfo ti, double *x, int N) {...}
```

- Rename thread management parameter

```
void library_func(Thread_Comm c, double *x, int N) {...}
```

Threads and Library Interfaces

Reflection

- Extra thread communication parameter

```
void library_func(ThreadInfo ti, double *x, int N) {...}
```

- Rename thread management parameter

```
void library_func(Thread_Comm c, double *x, int N) {...}
```

- Compare:

```
void library_func(MPI_Comm comm, double *x, int N) {...}
```

Threads and Library Interfaces

Reflection

- Extra thread communication parameter

```
void library_func(ThreadInfo ti, double *x, int N) {...}
```

- Rename thread management parameter

```
void library_func(Thread_Comm c, double *x, int N) {...}
```

- Compare:

```
void library_func(MPI_Comm comm, double *x, int N) {...}
```

Conclusion

- Prefer flat MPI over MPI+OpenMP for a composable software stack
- MPI automatically brings better data locality

OpenMP for GPUs

So WHY OpenMP for GPUs then??

OpenMP for GPUs

So WHY OpenMP for GPUs then??

- To support legacy FORTRAN applications

NVPROF

- Command line profiler for CUDA applications
- GUI profiler available as well: `NVVP`

```
  $> nvprof ./aa06697d.out
==32142== Profiling application: ./aa06697d.out
==32142== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   56.14%  2.0480us         1  2.0480us  2.0480us  2.0480us  transpose(double*, int)
                   23.68%    864ns         1    864ns    864ns    864ns  [CUDA memcpy HtoD]
                   20.18%    736ns         1    736ns    736ns    736ns  [CUDA memcpy DtoH]
      API calls:   69.47%  184.21ms         1  184.21ms  184.21ms  184.21ms  cudaMalloc
                   29.94%  79.390ms         1  79.390ms  79.390ms  79.390ms  cudaDeviceReset
                    0.37%  989.24us        94  10.523us  8.3810us  63.136us  cuDeviceGetAttribute
                   ....
```

CUDA MEMCHECK

- Valgrind-port for CUDA devices
- Subtools:
  1. `memcheck`: Check for invalid memory access or non-free'd memory
  2. `racecheck`: Check for race conditions when accessing shared memory
  3. `synccheck`: Check for properly placed synchronizations
  4. `initcheck`: Check for access to uninitialized device memory

```
  cuda-memcheck ./aa06697d.out
========= CUDA-MEMCHECK
========= Leaked 800 bytes at 0x7f9e86a00000
=========     Saved host backtrace up to driver entry point at cudaMalloc time
=========     Host Frame:/usr/lib/x86_64-linux-gnu/libcuda.so.1 (cuMemAlloc_v2 + 0x1b7) [0x2ba157]
=========     Host Frame:./aa06697d.out [0x385d3]
=========     Host Frame:./aa06697d.out [0xaf2b]
=========     Host Frame:./aa06697d.out [0x48b98]
=========     Host Frame:./aa06697d.out [0x672f]
=========     Host Frame:./aa06697d.out [0x6352]
=========     Host Frame:/lib/x86_64-linux-gnu/libc.so.6 (__libc_start_main + 0xe7) [0x21bf7]
=========     Host Frame:./aa06697d.out [0x60da]
=========
========= LEAK SUMMARY: 800 bytes leaked in 1 allocations
========= ERROR SUMMARY: 1 error
```

Profiling and Debugging

Reminder: Valgrind

- Checks for invalid memory accesses, non-free'd allocations, uninitialized memory, etc.
- Executes your code in a virtual environment with access monitoring
- Various subtools and options for deeper inspection
- Saves you A LOT of debugging time

```
  valgrind ./aa06697d.out
==32397== Memcheck, a memory error detector
==32397== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==32397== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==32397== Command: ./aa06697d.out
==32397==
==32397== 800 bytes in 1 blocks are definitely lost in loss record 83 of 94
==32397==    at 0x4C31B0F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==32397==    by 0x10E2E2: main (aa06697d.cu:37)
==32397==
==32397== LEAK SUMMARY:
==32397==    definitely lost: 800 bytes in 1 blocks
==32397==    indirectly lost: 0 bytes in 0 blocks
==32397==      possibly lost: 2,256 bytes in 15 blocks
==32397==    still reachable: 384,620 bytes in 78 blocks
==32397==         suppressed: 0 bytes in 0 blocks
==32397== Reachable blocks (those to which a pointer was found) are not shown.
==32397== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==32397==
==32397== For counts of detected and suppressed errors, rerun with: -v
==32397== ERROR SUMMARY: 16 errors from 16 contexts (suppressed: 0 from 0)
```

Profiling and Debugging

OpenCL: oclgrind

- Like `cuda-memcheck`, but for OpenCL
- No GPU required in order to use it
- Permissive BSD-style license, developed at the University of Bristol
- `https://github.com/jrprice/Oclgrind`

Exercises

Environment

- `https://gtx1080.360252.org/2020/ex9/`
- (Might receive visual updates and additional hints)
- Due: Tuesday, January 12, 2021 at 23:59pm

Hints and Suggestions

- Consider version control for locally developed code
- Please let me know of any bugs or issues