My experience with OpenMP off-loading C++ classes

Jean-Luc Fattebert

Oak Ridge National Laboratory
Context: ExaAM project, an ECP application (PI: John Turner, ORNL)

• What is Additive manufacturing?
  – The process of joining materials to make objects from 3D model data, usually layer upon layer, as opposed to subtractive manufacturing methodologies

• Goal
  – Improve quality, reliability, and application breadth of additive manufacturing for metallic alloys

• Computational approach
  – Coupling multiple codes modeling various length-scales
Fine-scale Microstructure using Phase-field model

Phase-field equation

\[ \frac{\partial \phi}{\partial t} = M \left[ \epsilon^2 \nabla^2 \phi + W \phi (1 - \phi) (1 - 2\phi) + \frac{\partial f}{\partial \phi} \right] \]

+ a few other coupled equations

C++11 Code: AMPE
https://github.com/LLNL/AMPE

solid  \rightarrow  liquid

• Case of binary alloy
  – At every mesh point of discretization grid, given $\phi$ (phase fraction) and $c$ (alloy composition), solve a set of nonlinear equations for $c_S$ and $c_L$ using a Newton solver

$$c = \phi c_S + [1 - \phi]c_L$$

$$\frac{\partial f^S}{\partial c_S} = \frac{\partial f^L}{\partial c_L}$$

• Ternary alloy
  – Similar with 4 unknowns and 4 equations

• $f^S$ and $f^L$ known functions, parameterized with 10+ parameters each
Solvers initial C++ implementation

• Base class
  – Newton solver
  – Pure virtual functions to compute right hand side and Jacobian

• Derived class
  – Implements specific right hand side and Jacobian computation

• CPU code

```cpp
#pragma omp parallel for
for(int i=0;i<N;i++)
{
    BinaryAlloySolver s(T[i]);
    double x[2];
    s.solve(phi[i],c[i], x);
    ...
}
```
What can I do or not do with OpenMP4.5 offload?

• Things I knew I could not use within OpenMP region
  – STL

• Things I suspected I could not use within OpenMP region
  – virtual functions
  – assert()

• Things I discovered I could not use within OpenMP region
  – Classes with non-trivial constructors/destructors, or even classes with declared a constructor and/or destructor
Strategy to offload code

• Platform: Summit @OLCF
• Compiler: gcc/10.2.0
• Compiler error messages not very specific/informative …
  – No info on which specific function or what cannot be offloaded
• Use “toy” code to test what the compiler let me do or not
  – Step-by-step move closer to target code design
Moving towards a working C++ code

• Remove STL, assert
• Make constructors/destructors trivial
  – Add setup functions to initialize objects
• Use Curiously Recurring Template Pattern (CRTP) to avoid virtual functions

Limited C++: C code + class encapsulation + templates
template <unsigned int Dimension, class SolverType>
Class NewtonSolver
{
    #pragma omp declare target
    void internalRHS(const double* const x, double* const fvec)
    {
        static_cast<SolverType*>(this)->RHS(x, fvec);
    }
    ...
    #pragma omp end declare target
}

class CALPHADConcSolverBinary : public NewtonSolver<2, CALPHADConcSolverBinary>
{
    public:
    #pragma omp declare target
    int ComputeConcentration(double* const conc, const double tol,
                          const int max_iters, const double alpha = 1.)
    {
        return NewtonSolver::ComputeSolution(conc, tol, max_iters, alpha);
    }
    #pragma omp end declare target
    ...
}
Driver code

```c
#pragma omp target
  map (to: sol ) map ( tofrom: xdev )
  map (to: fA, fB, Lmix_L, Lmix_A)
  map (to: RTinv), map ( from: nits)
{
  #pragma omp teams distribute parallel for
  for(int i=0;i<N;i++)
  {
    xdev[2*i]=sol[0];
    xdev[2*i+1]=sol[1];

    double hphi = 0.5+i*deviation;
    double c0 = 0.3;

    class Thermo4PFM::CALPHADConcSolverBinary solver;
    solver.setup(c0, hphi, RTinv, Lmix_L, Lmix_A, fA, fB);
    nits[i] = solver.ComputeConcentration(&xdev[2*i], 1.e-8, 50);
  }
}```
Performance

• 4.5X speedup GPU over CPU on Summit for ternary alloy problem (4 coupled equations)
  – 6 GPU offload vs. 42 OpenMP CPU threads

• Performance currently limited by GPU registers memory
  – Planning to replace some “double” with “float” to reduce memory requirements
Conclusion

• Working solution for gcc on Summit
  – Requires some non-trivial code changes
  – Still debugging some classes…

• Open source soon
  – https://github.com/ORNL/Thermo4PFM

• Decent performance, to be improved with mixed precision

• XL compiler not working for me at the moment

• Better (and user friendly) documentation about porting C++ classes would be really helpful
  – Probably dependent on compiler, compiler version,…

• More targeted error messages at compile time would help too…
Acknowledgements

• Research supported by the Exascale Computing Project (http://www.exascaleproject.org), a joint project of the U.S. Department of Energy’s Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation’s exascale computing imperative.

• This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725