EXPLORING OPENMP TARGET OFFLOADING FOR THE GRAPHCORE ARCHITECTURE

JOSE M MONSALVE DIAZ
Postdoctoral Researcher
Argonne National Laboratory
jmonsalvediaz@anl.gov

Esteban M Rangel
Sid Raskar
Johannes R Doerfert

November 9th, 2022
San José, CA
CAN WE FULLY SUPPORT OPENMP IN GRAPHCORE’S IPU ARCHITECTURE?
WHY?

This is a small exploratory project...

- Evaluate OpenMP’s ability for heterogeneity
- Evaluate using a dataflow-inspired architecture with OpenMP
- Allow scientific code to use AI accelerators in their code base:
  - OpenMP is already used for most applications
  - This could enable the use of surrogate models and similar
  - Can the IPU run both class of computation?
WHAT DO WE NEED FROM LLVM?

OpenMP compilation pipeline
1. BACKEND
LLVM-IR -> IPU Assembly

POPC COMPILER:
- LLVM Based
- Good documentation
- But, closed source:
  - Using driver options, we discover it possible to create simple vertex
  - We can bypass the front end
2. FRONTEND
OpenMP -> LLVM-IR

We need to represent

- Vertex (Codelets)
- Compute Set
- Tensors
- IPU program
- Mapping to IPU tiles
2. FRONTEND
OpenMP -> LLVM-IR

Vertex

```c
#pragma omp task depend(in:...) depend(out:...) {
    /// Codelet compute function
}
```
2. FRONTE ND
OpenMP -> LLVM-IR

Compute set

```c
#pragma omp parallel num_threads(num_tiles)
{
    #pragma omp masked filter(…)
    #pragma omp task depend(in:…) depend(out:…)

    #pragma omp masked filter(…)
    #pragma omp task depend(in:…) depend(out:…)
}
```
2. FRONTEND
OpenMP -> LLVM-IR

Tensors

```c
int v1, v2;
float *t1, *t2;
#pragma omp target map( t1[0:10][0:20], t2[10], v1, v2)
{
    float t3[30][30];
}
```
while (v1 < v2) { 
copy (t1, t2);
} 
if (v3 < v4) 
copy (t2, t3);
else 
copy (t3, t2);
}
3. PLUGIN
OpenMP→Poplar

- Using the poplar library

- Modifications will be required for exposing the IPU program API in poplar
  - Plugin has functions tailored for SPMD execution mode
4. DRIVER

Gluing it all together
THANKS!

COMMENTS QUESTIONS CONCERNS:

jmonsalvediaza@anl.gov