LLVM DEVELOPERS' MEETING 2022

Argonne NATIONAL LABORATORY

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?

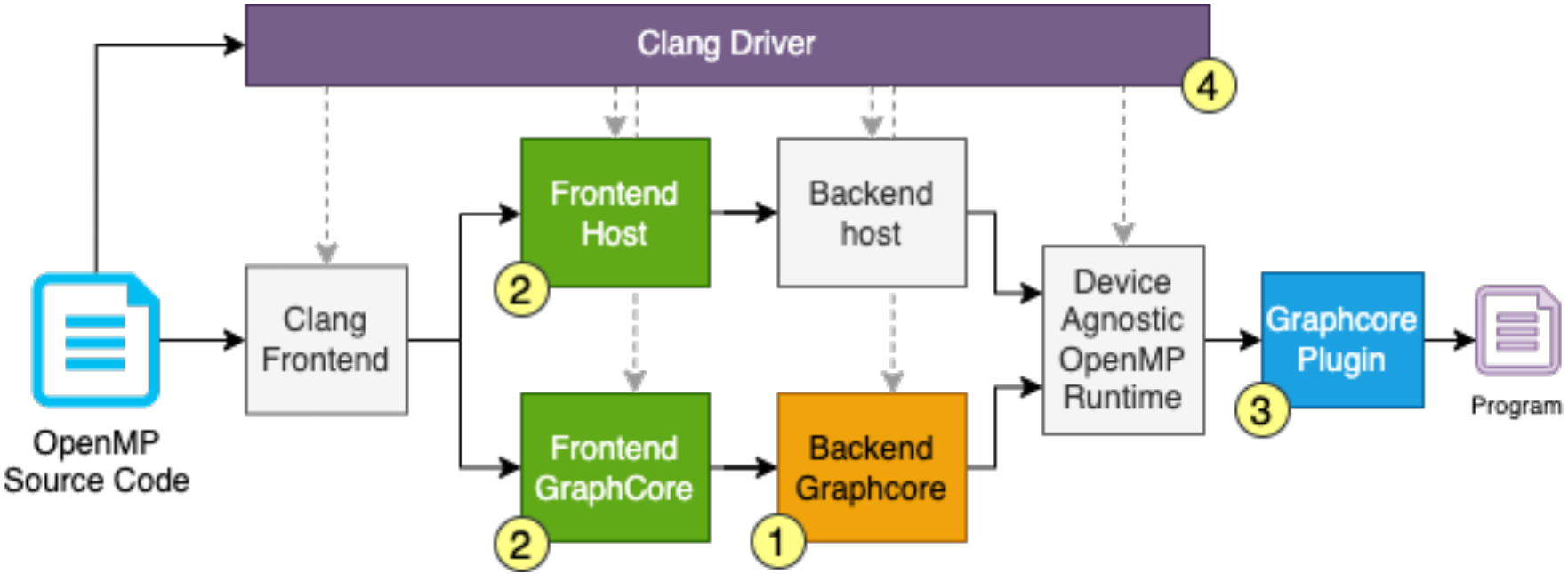
WHY?

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?

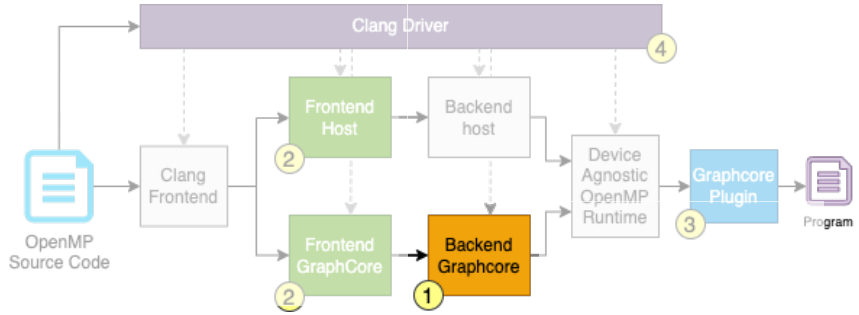
WHAT DO WE NEED FROM LLVM?

OpenMP compilation pipeline



1. BACKEND

LLVM-IR -> IPU Assembly



POPC COMPILER:

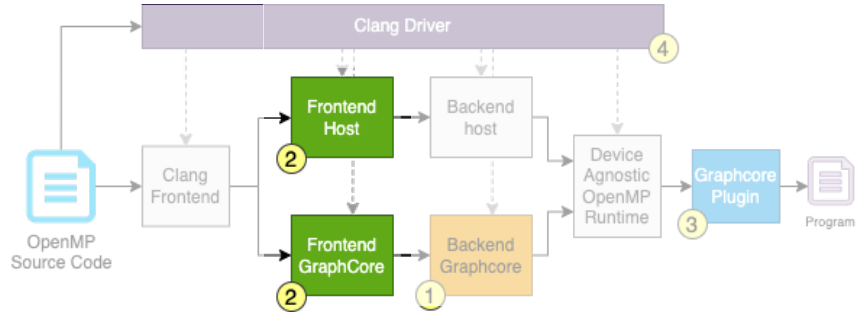
- LLVM Based
- Good documentation
- But, closed source:
  - Using driver options, we discover it possible to create simple vertex
  - We can bypass the front end

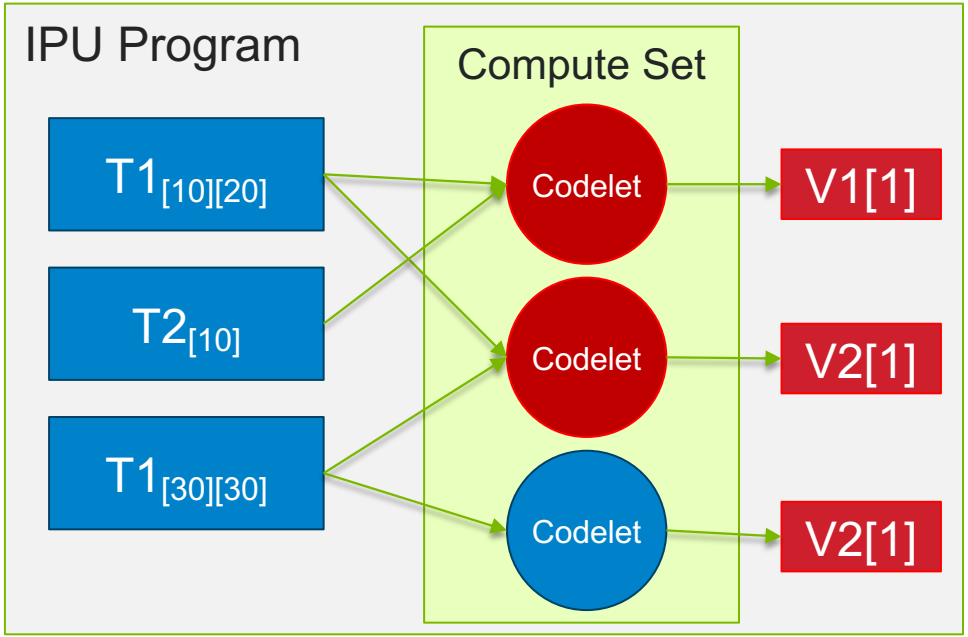
2. FRONTEND

OpenMP -> LLVM-IR

We need to represent

- Vertex (Codelets)
- Compute Set
- Tensors
- IPU program
- Mapping to IPU tiles

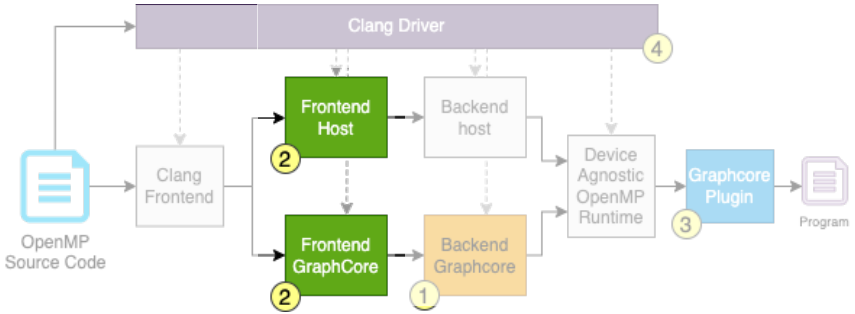




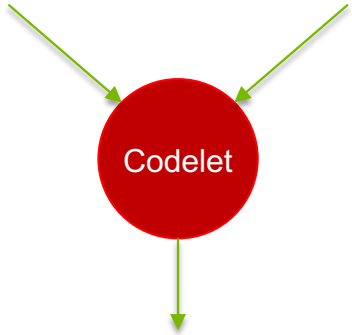
2. FRONTEND

OpenMP -> LLVM-IR

Vertex

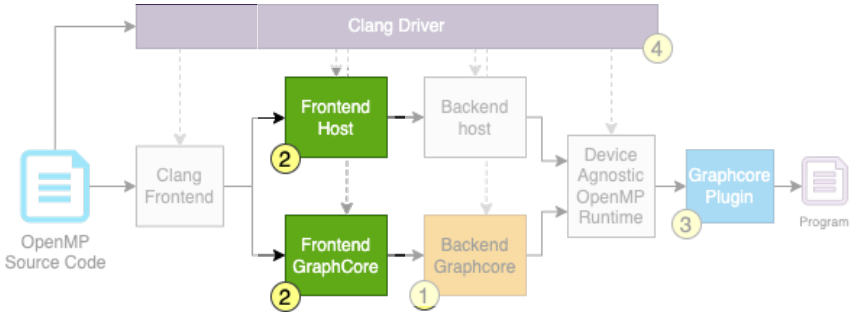



```
#pragma omp task depend(in:…) depend(out:…)
{
   /// Codelet compute function
}
```



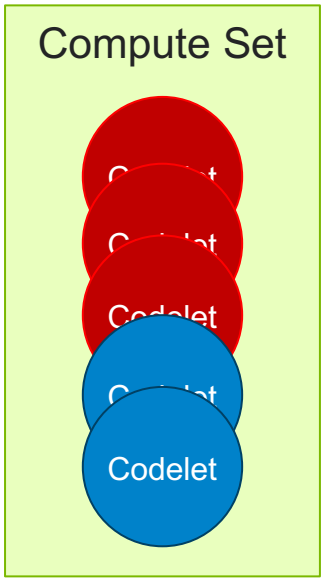
2. FRONTEND

OpenMP -> LLVM-IR



Compute set

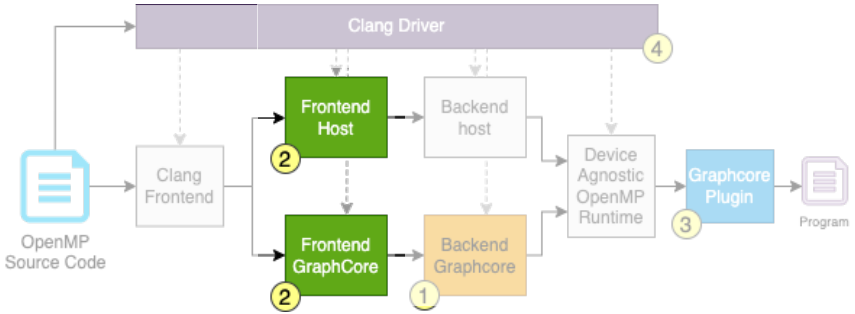
```
#pragma omp parallel num_threads(num_tiles)
{
#pragma omp masked filter(…)
  #pragma omp task depend(in:…) depend(out:…)

#pragma omp masked filter(…)
  #pragma omp task depend(in:…) depend(out:…)
}
```



2. FRONTEND

OpenMP -> LLVM-IR

Tensors




```
int v1, v2;
float *t1, *t2;
#pragma omp target map( t1[0:10][0:20], t2[10], v1, v2)
{
   float t3[30][30];
}
```

$T1_{[10][20]}$

$T2_{[10]}$

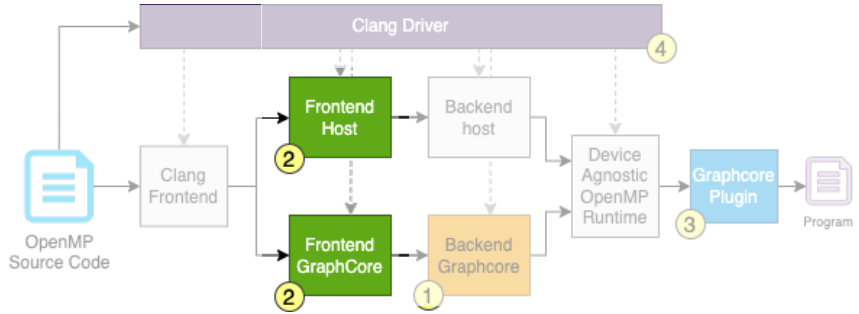
$T1_{[30][30]}$

V1[1]

V2[1]

FRONTEND

OpenMP -> LLVM-IR



IPU Program

```
#pragma omp target map(…)
{
  while(v1 < v2) {
    copy(t1, t2);
    #pragma omp parallel
    {… Compute set description …}
    if (v3 < v4)
      copy(t2, t3);
    else
      copy(t3, t2);
    #pragma omp parallel
    {… Compute set description …}
  }
}
```

Control flow

Intrinsic functions

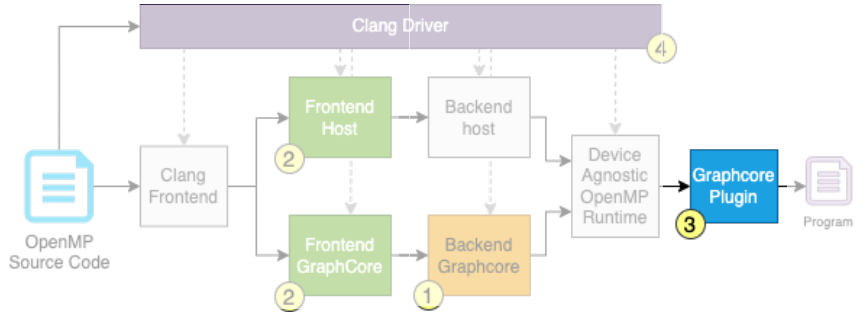
OpenMP Regions

```
while(v1 < v2)
  copy(t1, t2)
  execute(CS1)
  if (v3 < v4)
    copy(t2, t3)
  else
    copy(t3, t2)
  execute(CS2)
```

IPU Program

3. PLUGIN

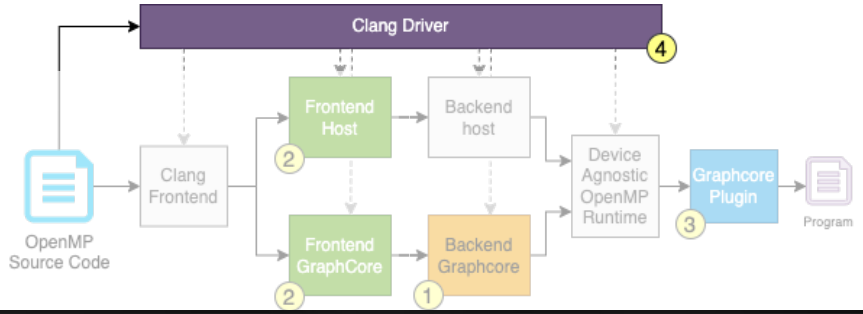
OpenMP→Poplar



- 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




```
jmonsalvediaz@graphcore:~/installers/clang/tests$ clang -ccc-print-bindings a.c -fopenmp -fopenmp-targets=colossus
# "x86_64-unknown-linux-gnu" - "clang", inputs: ["a.c"], output: "/tmp/a-c7312b.bc"
# "colossus-graphcore-unknown-elf" - "clang", inputs: ["a.c", "/tmp/a-c7312b.bc"], output: "/tmp/a-d6b0bc.ll"
# "colossus-graphcore-unknown-elf" - "COLOSSUS::Backend", inputs: ["/tmp/a-d6b0bc.ll"], output: "/tmp/a-303e02.s"
# "colossus-graphcore-unknown-elf" - "COLOSSUS::Backend", inputs: ["/tmp/a-303e02.s"], output: "/tmp/a-0e0635.o"
# "x86_64-unknown-linux-gnu" - "Offload::Packager", inputs: ["/tmp/a-0e0635.o"], output: "/tmp/a-f77012.out"
# "x86_64-unknown-linux-gnu" - "clang", inputs: ["/tmp/a-c7312b.bc", "/tmp/a-f77012.out"], output: "/tmp/a-8dacec.o"
# "x86_64-unknown-linux-gnu" - "Offload::Linker", inputs: ["/tmp/a-8dacec.o"], output: "a.out"
jmonsalvediaz@graphcore:~/installers/clang/tests$ clang -ccc-print-phases a.c -fopenmp -fopenmp-targets=colossus
            +- 0: input, "a.c", c, (host-openmp)
          +- 1: preprocessor, {0}, cpp-output, (host-openmp)
       +- 2: compiler, {1}, ir, (host-openmp)
       |              |     +- 3: input, "a.c", c, (device-openmp)
       |              |  +- 4: preprocessor, {3}, cpp-output, (device-openmp)
       |              |- 5: compiler, {4}, ir, (device-openmp)
       |           +- 6: offload, "host-openmp (x86_64-unknown-linux-gnu)" {2}, "device-openmp (colossus-graphcore-unknown-elf)" {5}, ir
       |        +- 7: colossus-popc-compiler, {6}, assembler, (device-openmp)
       |     +- 8: colossus-popc-compiler, {7}, object, (device-openmp)
       |  +- 9: offload, "device-openmp (colossus-graphcore-unknown-elf)" {8}, object
       |- 10: clang-offload-packager, {9}, image
    +- 11: offload, "host-openmp (x86_64-unknown-linux-gnu)" {2}, " (x86_64-unknown-linux-gnu)" {10}, ir
 +- 12: backend, {11}, assembler, (host-openmp)
+- 13: assembler, {12}, object, (host-openmp)
14: clang-linker-wrapper, {13}, image, (host-openmp)
```

THANKS!

COMMENTS QUESTIONS CONCERNS:

jmonsalvediaz@anl.gov