OpenMP Accelerator Support for GPU

Kelvin Li (kli@ca.ibm.com)
IBM Canada Lab

Revolutionizing the Datacenter
Outline

- OpenMP API
- OpenMP accelerator support
- OpenMP accelerator support in IBM XL Compiler
OpenMP API

- A de-facto industry standard for parallel programming for the last 15+ years
- Supports Fortran and C/C++ programming languages
- Supports shared-memory and accelerator programming model
- ARB consists of many vendors, national labs and universities

About the OpenMP ARB and OpenMP.org

Mission
The OpenMP mission is to standardize directive-based multi-language high-level parallelism that is performant, productive and portable. Jointly defined by a group of major computer hardware and software vendors and major parallel computing user facilities, the OpenMP API is a portable, scalable model that gives shared-memory parallel programmers a simple and flexible interface for developing parallel applications on platforms ranging from embedded systems and accelerator devices to multicore systems and shared-memory systems. The OpenMP ARB owns the OpenMP brand, oversees the OpenMP specification and produces and approves new versions of the specification.
OpenMP accelerator support

- OpenMP API
  - V4.0 published in 2013 to include the accelerator support
  - V4.5 published in 2015 to extend the accelerator support
  - continue to work on more advance features ...

- The execution model is host-centric
  - The host device offloads target regions to the target devices
  - The target devices can be GPU, DSP, coprocessor etc.

- Insert directives to the code block that is offloaded to the device

- Clauses of the directives to control data movement between the host device and the target device
OpenMP accelerator support

- Host-centric: host device offloads target regions (code and data) to target devices
- The target pragma/directive encloses the code block that is executed on the device
- The code is offloaded to be executed on the target device
- This model allows the target region executed on the host device

```c
#define N 500
float z[N], y[N], x[N];
float a;
...
#pragma omp target
{
  for (i=0; i<N; i++)
    z[i] = a*x[i] + y[i];
}
...
```

```c
for (i=0; i<N; i++)
  z[i]=a*x[i]+y[i];
```
OpenMP accelerator support

- The teams construct creates a league of thread teams
- Exploit extra level of parallelism on some hardware (e.g. NVIDIA GPU)
  - Thread blocks in CUDA
- Furthermore, work can be distributed among the teams using the distribute construct
- The iteration of the for loop is chunked and distributed among the teams

```c
#pragma omp target
#pragma omp teams
#pragma omp distribute
for (i=0; i<N; i++)
    z[i] = a*x[i] + y[i];
```
OpenMP accelerator support

- The `distribute parallel for` construct exploits the parallelism among the teams and within each team.
- The iterations of the for loop are chunked and distributed among the teams.
- Each chunk is executed in parallel within the teams.

```
#pragma omp target
#pragma omp teams
#pragma omp distribute parallel for
for (i=0; i<N; i++)
    z[i] = a*x[i] + y[i];
```
OpenMP accelerator support

- The spec also provides data mapping clauses to control data movement between the host device and the target device
OpenMP accelerator support in XL compiler

- XL C/C++ and XL Fortran compilers comply with OpenMP V3.1
- IBM compiler team experiments the accelerator support in the XL compiler targeting NVIDIA GPU
- Collaborate with IBM research team
- Collaborate with the LLVM/Clang community for common runtime interface
- Work closely with NVIDIA engineers to access GPU hardware technology
- Build on existing advanced optimization technology in the XL compiler
- Aim at optimized code gen for both CPU and GPU
  - Fully exploit the POWER8 and Nvidia GPU hardware
OpenMP 4.0/4.5 – IBM XL Compiler Architecture

C/C++ Source code (OpenMP directives) → XL C/C++ Front End → High-Level Optimizer

- Outlining/Duplication to separate Host + Accelerator targets
- High Level Optimizer (host + target)

POWER Low-level Optimizer
- Low-level Optimizations
- Register Allocation + Scheduling
- POWER Code Generation

C/C++ Source code (OpenMP directives) → XL Fortran Front End → High-Level Optimizer

Fortran Source code (OpenMP directives)

Wcode (+OpenMP) → CPU/GPU Partitioner

CPU Wcode → POWER Low-level Optimizer

- Libraries
- Profile Information

GPU Wcode (omp target regions) → libNVVM

Wcode to LLVM IR translator
- NVVM IR

NVVM = LLVM with NVIDIA enhancements

OpenMP GPU code generation (omp target)

OpenMP GPU Run Time

CUDA Runtime

CUDA Driver

System Linker

Executable for POWER/GPU system

NVPTX CodeGen

PTXAS

nvlink

libNVVM

OpenMP GPU Front End

XL Fortran Front End

XL C/C++ Front End
Summary

- Prototype basic target constructs and data mapping clauses
- Come to visit the IBM XL Compiler booth to check out what we have
- We are looking for sponsored users, if you are interested, let us know
Questions
Thank You
Additional Information

- XL C/C++ home page

- C/C++ Café
  http://ibm.biz/Bdx8xR

- XL Fortran home page

- Fortran Café
  http://ibm.biz/Bdx8XX