1. Introduction

- Directive-based programming is an emerging approach for reducing development efforts needed to implement parallel code on an accelerator device such as the graphics processing unit (GPU) and Xeon Phi. Accelerated code can be easily obtained by adding OpenACC directives to the serial code.
- The application code can be easily rolled back to the original CPU code, which does not include accelerator-specific description. From a long-range point of view, this type of code flexibility is essential to achieve performance portability, which ensures high-performance on different, future platforms.

2. Motivation

- Rewriting of OpenACC code is needed to solve the same problem size as that processed by CPU-based systems, because the capacity of device memory is an order of magnitude smaller than that of main memory.
- Portability is a critical concern for developers of large data applications, and it is difficult to achieve high performance on multiple devices with a single source code.

3. Goal

- Our goal is to realize a directive-based programming framework capable of accelerating large-scale computation on a GPU.
- Our method for achieving this goal is an extension of OpenACC, named pipelined accelerator (PACC). We also develop a source-to-source translator that automates the key procedure:
  - Data division for fitting large data into device memory
  - Pipelined execution with overlapping data transfer with computation

4. Related work

- Sabne et al. [1] processed large data by allowing OpenMP-like programs to run on multiple GPUs. Their translator automatically divides and transfers data, which is overlapped with kernel execution. In contrast to this OpenMP-based approach, PACC is based on OpenACC designed for accelerators.
- XcalableMP [2] is a directive-based language extension designed for distributed memory systems. Although PACC cannot distribute data over nodes, it can increase the maximum data size that can be solved on each node.

5. PACC: Pipelined ACCELERator

- PACC directives allow application developers to specify a data division scheme and an execution configuration of the pipeline.
- According to this specification in the PACC code, our PACC translator generates pipelined OpenACC code. The following key rewrite rules are automated using the ROSE compiler framework (http://rosecompiler.org/):
  - Data division and buffer allocation
  - Rewriting of loop structures and data references
  - Replacement of PACC directives to OpenACC directives

6. Example of PACC code

- 5-point stencil computation processed in a pipeline
  - 2D arrays s and d are divided into chunks of N elements

```
pragma acc parallel loop copyin(s[0:n]) copyout(d[1:n-2])
for (i = 1; i < n-1; i++)
  d[i] = s[i] + s[i+1] + s[i+2];
```

```
int csize = N / threadblockSize;
for (j = 1; j < csize; j++)
  for (i = 0; i < csize-1; i++)
    d[csize*j+i] = s[csize*j+i] + s[csize*j+i+1];
```

- The generated code divides large data into small chunks, which are then processed in pipeline.
- Each chunk is stored in a buffer separated from the original large data. The original data must be separated from a chunk, because OpenACC 1.0 assumes the same variable name between host data and device data.

7. Evaluation

- Experimental Applications
  - Black Scholes [3]
  - Sobel Filter [3]
  - 27-point stencil [4]

- Experimental machine
  - Intel Core i7 3930K (6 cores, 32 GB)
  - NVIDIA Tesla K20 (5 GB)
  - Windows 7 64bit
  - PGI compiler 14.4, CUDA 6.0

- We implemented an alternative version that used acc_map_data, a new capability of OpenACC 2.0, to eliminate such a buffer from main memory. This mapping function increased the maximum data size to 29 GB, but it disabled pinned memory, which realized pipelined execution. Consequently, the performance was slightly lower than the non-pipelined version.

8. Conclusion

- We presented PACC, an extension of OpenACC, which realizes pipelined processing of large data on a GPU without changing the structure of the serial code. Application developers can process large data by adding some PACC directives to the serial code.
- Our extended directives increased the maximum data size by an order of magnitude. The maximum data size is determined by the capacity of main memory rather than that of device memory. The speedups over OpenMP implementation ranged from a factor of 2.6 to that of 8.7.

---

[4] [http://www.xrce.osaka-u.ac.jp](http://www.xrce.osaka-u.ac.jp/)

This study was partly supported by the JST CREST program “An Evolutionary Approach to Construction of a Software Development Environment for Massively-Parallel Computing Systems” (http://www.xrce.osaka-u.ac.jp/).