Benchmarking and Evaluating Unified Memory for OpenMP GPU Offloading

Alok Mishra¹, Lingda Li², Martin Kong², Hal Finkel³, Barbara Chapman^{1,2}

¹Stony Brook University ²Brookhaven National Laboratory ³Argonne National Laboratory Nov 13th, 2017 LLVM-HPC 2017, Denver, CO



A CENTURY OF SERVICE

NATIONAL LABORAT

* Stony Brook University Argonne

Important points

Detailed contents

Benchmarking and Evaluating Unified Memory for OpenMP GPU Offloading

Alok Mishra¹, Lingda Li², Martin Kong², Hal Finkel³, Barbara Chapman^{1,2}

¹Stony Brook University ²Brookhaven National Laboratory ³Argonne National Laboratory Nov 13th, 2017 LLVM-HPC 2017, Denver, CO



A CENTURY OF SERVICE

NATIONAL LABORA'



OpenMP GPU Offloading

- GPU is increasingly important in HPC
 - Massive threading capability
 - Energy efficient
- OpenMP 4.X offers GPU programming ability
- Compared with native models (CUDA, OpenCL)
 - Easy to learn, better performance portability
- Compared with other directive based models (e.g., OpenACC)
 - Boarder user community, better compiler support

We expect more developers will use OpenMP to program GPUs

NATIONAL LABORATORY

DF

Unified Memory

U.S. DEPARTMENT OF



DF

A CENTURY OF SERVICE

- Recent GPU architectures introduce enhanced support for unified memory
 - CPU and GPU use a single unified address space
 - On-demand page migration, cache coherence in Volta
- Unified memory facilitates GPU programming
 - Simplify hierarchical data structure copy (deep copy)
 - Enable GPU memory oversubscription

NATIONAL LABORATORY

No comprehensive study for unified memory yet

This paper aims to study the performance of unified memory under OpenMP

Key Problems

• Little effort has been put into OpenMP GPU offloading benchmarking

1. Need to develop a set of OpenMP GPU benchmarks for performance evaluation

 There is no official unified memory support in the current OpenMP yet

2. Need to implement a lightweight way to support unified memory for the current OpenMP







OpenMP Offloading without Unified Memory



OpenMP Offloading with Unified Memory



- Challenge 1: GPU memory allocation is not cneeded for LLVM the unified memory space
 - Solution: modify omp_target_alloc to allocate data in unified memory space
- Challenge 2: OpenMP runtime transfers data explicitly
 - Solution: use is <u>device</u> ptr to let OpenMP runtime step down from memory management







Example

```
#pragma omp target data map(to: A, B) map(from: C)
{
#pragma omp target teams distribute
for (int i = 0; i < N; i++) {
    #pragma omp parallel for
    for (int j = 0; j < M; j++)
        C[i][j] = A[i][j] + B[i][j];
}
</pre>
```

 Implement both traditional offloading and unified memory versions for most benchmarks

NATIONAL LABORATORY



A CENTURY OF SERVICE

Deep Copy & Unified Memory

- Hierarchical data structure mapping
 - Map the current instances and all indirectly referenced data
 - Programmers' burden
 - Time consuming, error prone
- TR6 will introduce custom mapper
 - Alleviate deep copy
- Unified memory solves deep copy (perfectly?)
 - Indirectly referred data are moved on demand

typedef struct {
 int input_n;
 int hidden_n;
 int output_n;
 float *input_units;
 float *hidden_units;
 float *hidden_delta;
 float *hidden_delta;
 float *target;
 float **input_weights;
 float **hidden_weights;
 float **hidden_prev_weights;
 float **hidden_prev_weights;
} BPNN;







Experimental Methodology

- Hardware
 - SummitDev @ ORNL
 - Tesla P100 NVLink
 - POWER 8
- Software
 - IBM Clang/LLVM with OpenMP GPU offloading support
 - CUDA 8.0
- Benchmarks
 - Backprop, BFS, CFD, K-means, NN, SRAD
 - https://gitlab.com/alokmishra.besu/rodinia_benchmark
 - Evaluate performance with different input sizes

CPU	2 POWER8
Cores / Socket	10
Threads / Core	8
CPU Clock	$2.0 \mathrm{GHz}$
Main Memory	256GB DDR4
GPU	4 Tesla P100
SMs / GPU	56
SM Clock	1328 MHz
Register File Size / SM	256 KB
FP32 CUDA Cores / SM	64
FP64 CUDA Cores / SM	32
GPU L2 Cache Size	4096 KB
GPU Memory	16GB HBM2
CPU & CPU Interconnect	NVL ink 10

Value

Parameter







Performance Results: CPU vs. GPU



- Computational bound benchmarks prefer GPU
- Memory bound benchmarks prefer CPU

NATIONAL LABORATORY

U.S. DEPARTMENT OF

Most time is used to transfer data between CPU and GPU



Performance Results: GPU w/o UM vs. GPU w/ UM (1)



- Group 1: benchmarks with lots of data reuse
 - On demand paging amortizes data movement overhead
 - Unified memory suffers from significant performance degradation when GPU memory is oversubscribed







Performance Results: GPU w/o UM vs. GPU w/ UM (2)





- Group 2: benchmarks with little data reuse
 - The performance is roughly proportional to input size
 - GPU w/o UM performs slightly better thanks to lower runtime overhead







Unified Memory Analysis



Data transfer



On demand paging introduces extra overhead

The overhead of Group 1 increases significantly in case of memory oversubscription

DF RY

CE

Improve Unified Memory Performance

- Applications with significant data reuse
 - Avoid/reduce data thrashing
 - Pin data with good locality into GPU memory
 - Pinned data cannot be thrashed by poor locality data
 - Use traditional data mapping to achieve this
- Applications with little data reuse
 - Reduce the runtime overhead associated with unified memory: page faults, on demand data transfer, ...
 - Use prefetching (e.g., cudaMemPrefetchAsync)







Conclusion

- Unified memory has many advantages
 - Enable GPU memory oversubscription
 - Address deep copy well
 - Ease to use under the current/future OpenMP standard
- Applications with little reuse
 - Unified memory performs slightly worse
 - Reduce on demand paging overhead
- Applications with large amounts of reuse
 - Unified memory can bring better performance
 - Be aware of data thrashing under memory oversubscription
- Benchmarks are available @ <u>https://gitlab.com/alokmishra.besu/rodinia_benchmark</u>







Thanks! Questions?

- Supported by ECP SOLLVE (OpenMP project)
- Find more about SOLLVE
 - Martin Kong's talk at OpenMP booth (1246), 11:15am, Wednesday
 - Barbara Chapman's talk at DOE booth (613), 2:30pm, Wednesday
- Find more about unified memory and OpenMP
 - Lingda Li's demo at DOE booth (613), 4:00pm, Wednesday





