

DATA TRANSFER AND REUSE ANALYSIS TOOL FOR GPU-OFFLOADING USING OPENMP

Alok Mishra¹, Abid M. Malik² and Barbara Chapman^{1,2}

¹Stony Brook University - USA, ²Brookhaven National Laboratory - USA



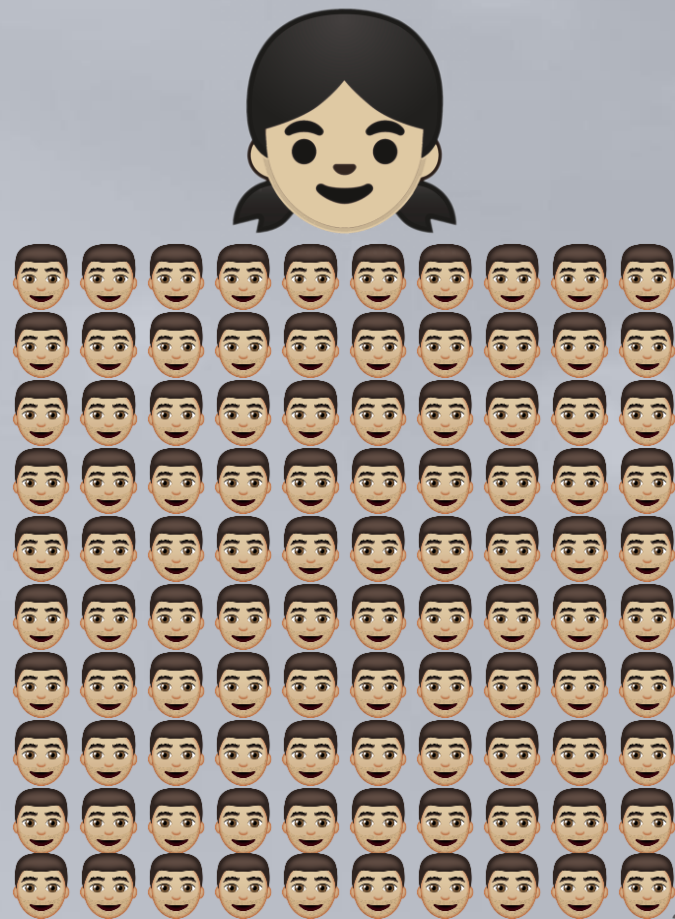
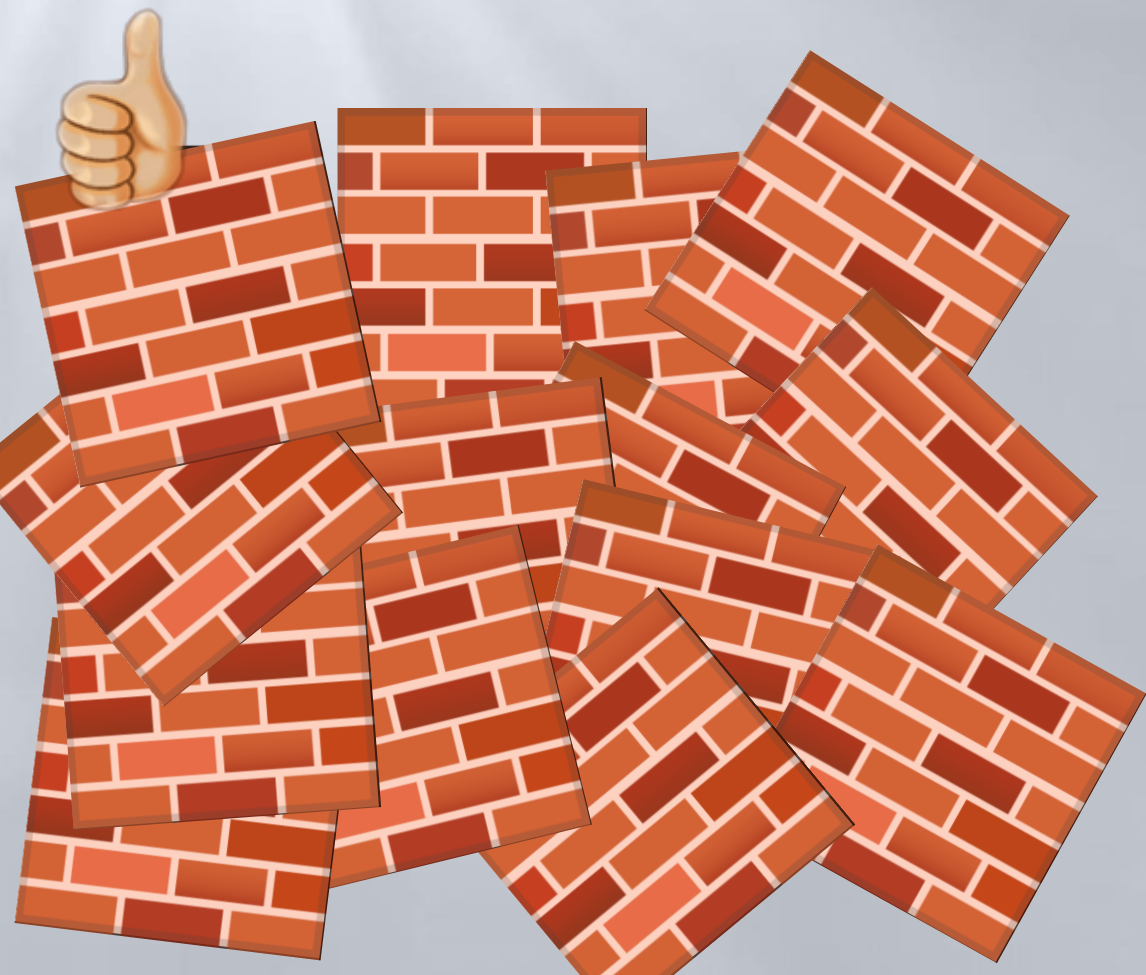
MOTIVATION

- ❖ HPC programs are parallel
- ❖ Long execution time
- ❖ Large size
- ❖ GPUs are increasingly important in HPC
 - Massive threading capability
 - Energy efficient



*A race car can travel faster,
but a bus can carry more load*





CHALLENGES OF PORTING TO GPU



Portability

Highly dependent on underlying architecture and choice of programming model (CUDA)



Programmability

Different from existing programming languages. Extensive refactoring of code is required



Parallelism

What is the degree of parallelism?

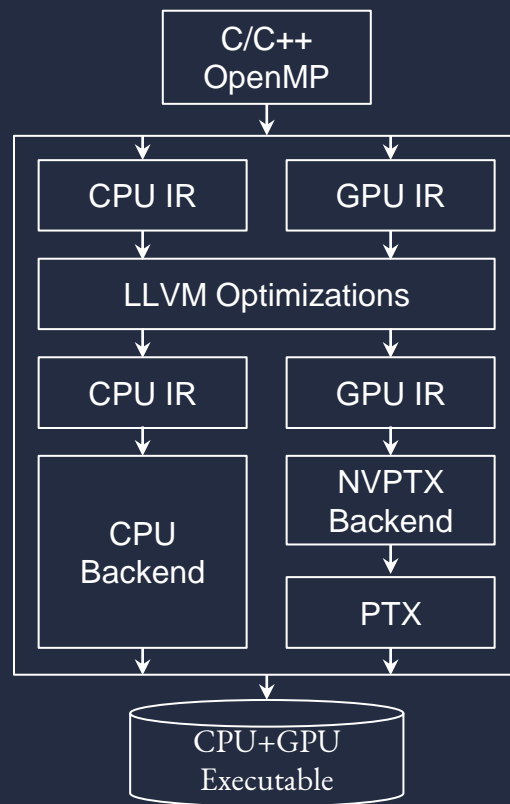


Data Handling

Requires explicit data transfers

- ❖ De-facto programming model for node-level parallelism
- ❖ OpenMP 4.X+ offers GPU programming ability
- ❖ OpenMP codes may also spend a significant portion of their execution time on data transfer
- ❖ Multiple GPU kernel calls may be reusing the same data

OpenMP



Data Transfer

Explicit

- target data map directive

Implicit

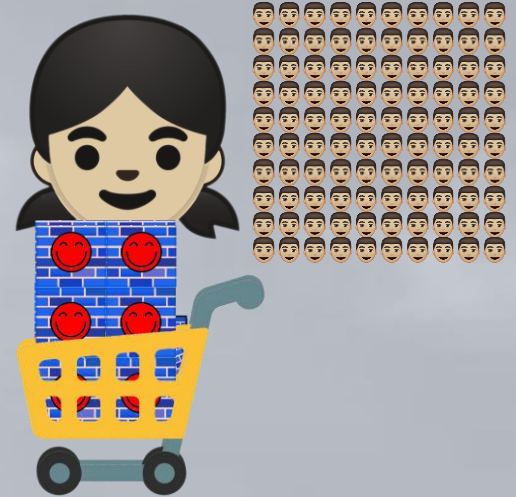
- transfer all data *to/from*
- compilers handle transfer



1. Color bricks in Blue

2. Draw smiley faces

3. Stack the bricks






**KEEP
CALM**

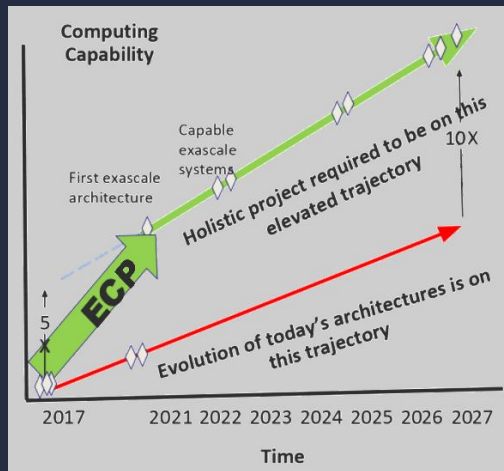
AND SEE THE

**BIGGER
PICTURE**

A light gray world map centered on the Atlantic Ocean, serving as a background for the text on the right side of the slide.

**Program
Transformation
for automatic GPU-
offloading
using
OpenMP**

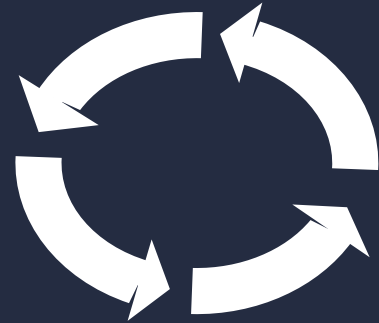
Exascale Computing Project



- ❖ Major US DoE* project
- ❖ Deliver 2 capable Exascale system
- ❖ Exaflop/s rate is 10^{18} floating point operations per second
- ❖ Acceptable power
- ❖ Develop applications to utilize them
- ❖ Develop softwares to make them usable
- ❖ Grid – Data Parallel Math library in C++

- ❖ Design & develop a compiler framework for C/C++ that can automatically
 - Recognize data reuse opportunities in an application
 - Insert pertinent “*omp data map*” directive in the source code accordingly.
 - Assumption - target offloading code already inserted

DATA REUSE ANALYSIS

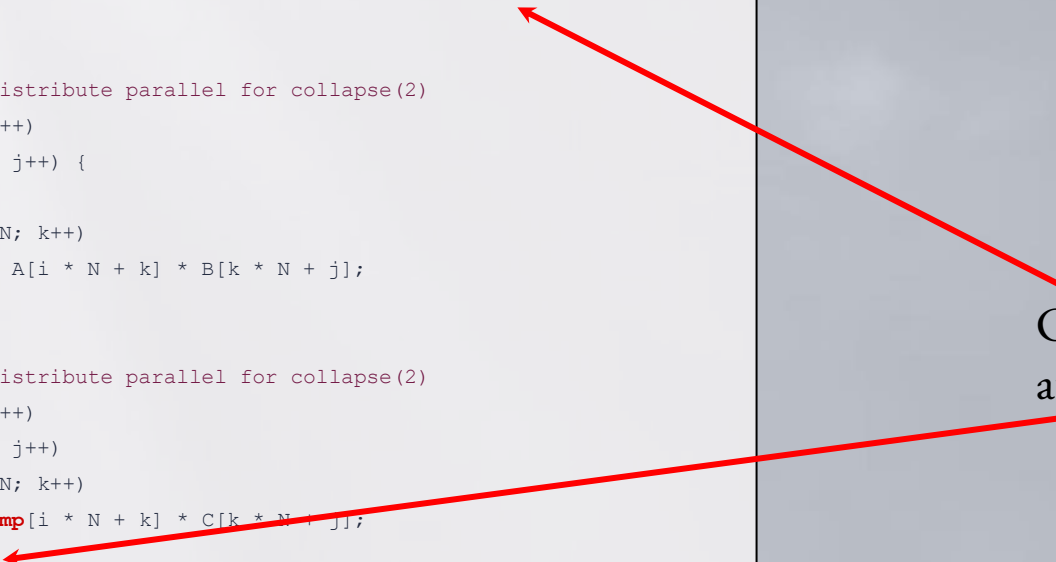


EXAMPLE CODE: MULTIPLY 3 MATRICES

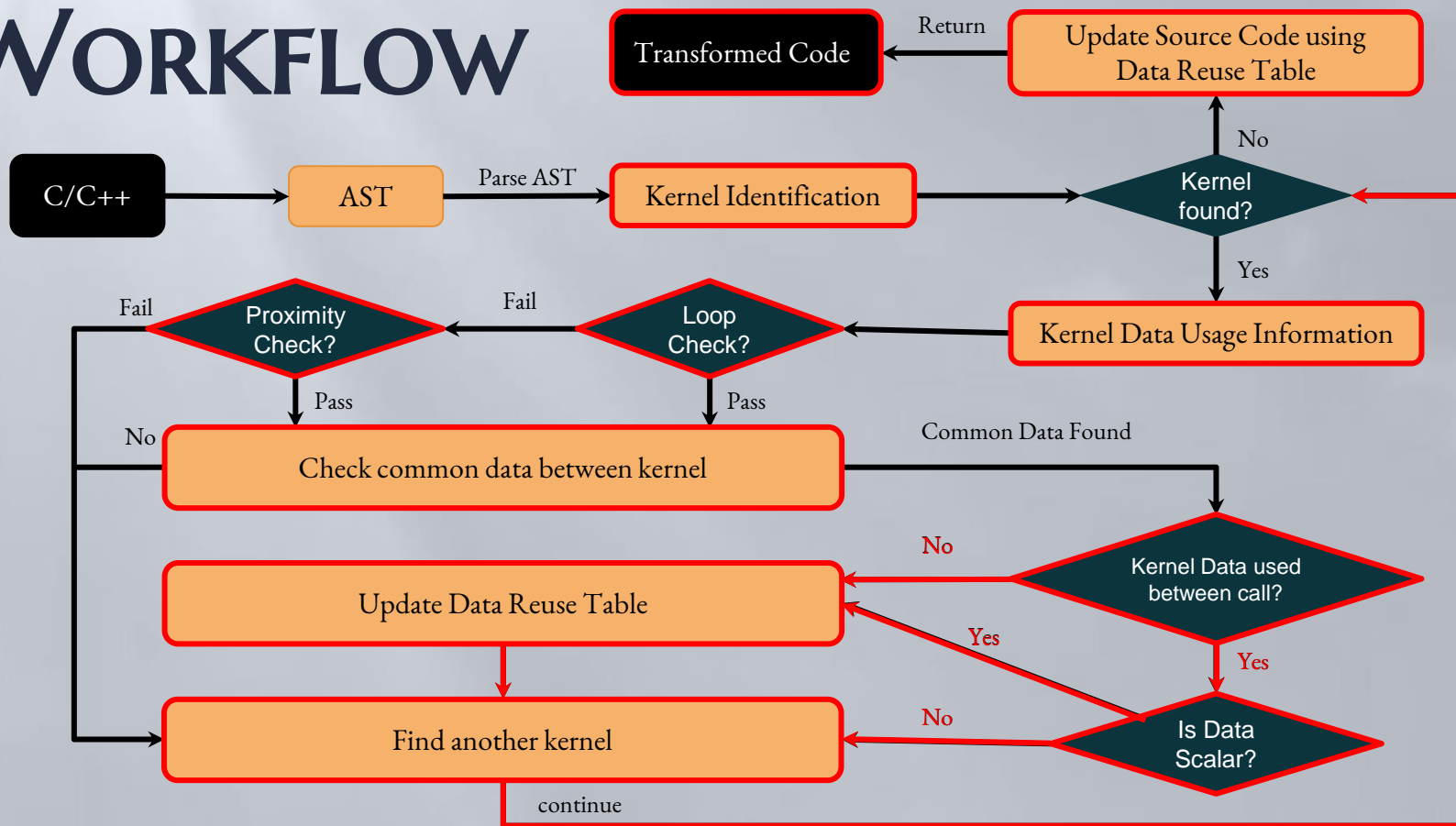
```
#pragma omp target data map(to: A[0:N*N], B[0:N*N], C[0:N*N]) \
                        map(tofrom: D[0:N*N]) map(alloc: temp[0:N*N])

{ // data region starts
// Kernel 1
#pragma omp target teams distribute parallel for collapse(2)
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++) {
            temp[i * N + j] = 0;
            for (int k = 0; k < N; k++)
                temp[i * N + j] += A[i * N + k] * B[k * N + j];
        }
// Kernel 2
#pragma omp target teams distribute parallel for collapse(2)
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            for (int k = 0; k < N; k++)
                D[i * N + j] += temp[i * N + k] * C[k * N + j];
} // data region ends
```

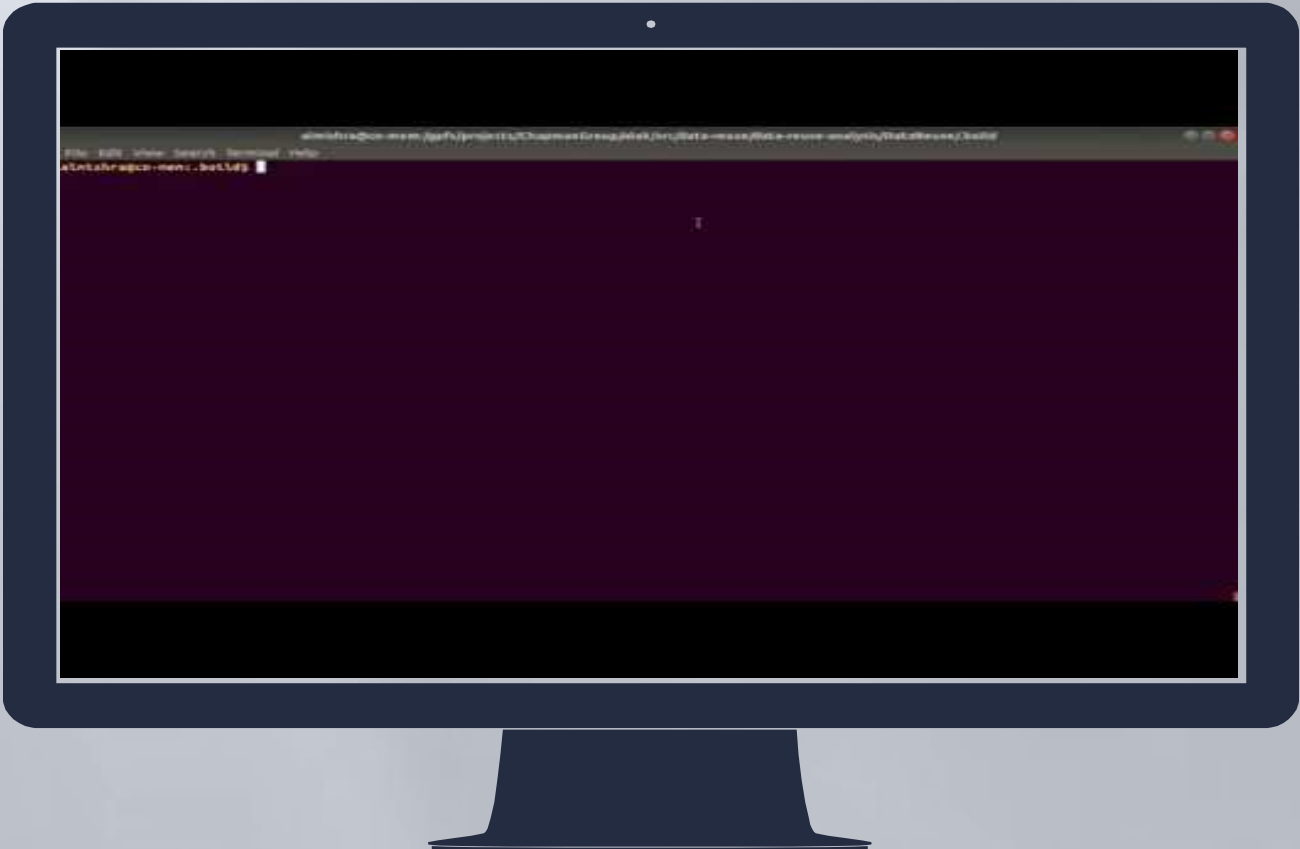
Code inserted
automatically



WORKFLOW



DEMO



EXPECTATION

```
#pragma omp target data map(to: A, B, C) \
                        map(tofrom: D) map(alloc: temp)
{
#pragma omp target teams distribute parallel for
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < N; k++)
                temp[i][j] += A[i][k] * B[k][j];
        }

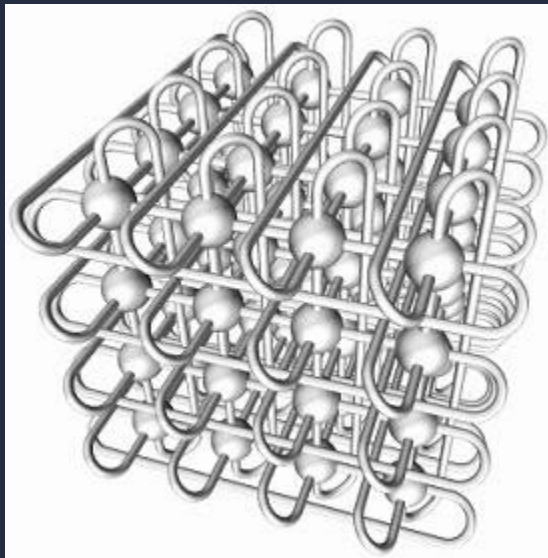
#pragma omp target teams distribute parallel for
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            for (int k = 0; k < N; k++)
                D[i][j] += temp[i][k] * C[k][j];
}
```

REALITY

```
#pragma omp target data map(to:temp)
{
#pragma omp target data map(to:A,B)
#pragma omp target teams distribute parallel for
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < N; k++)
                temp[i][j] += A[i][k] * B[k][j];
        }

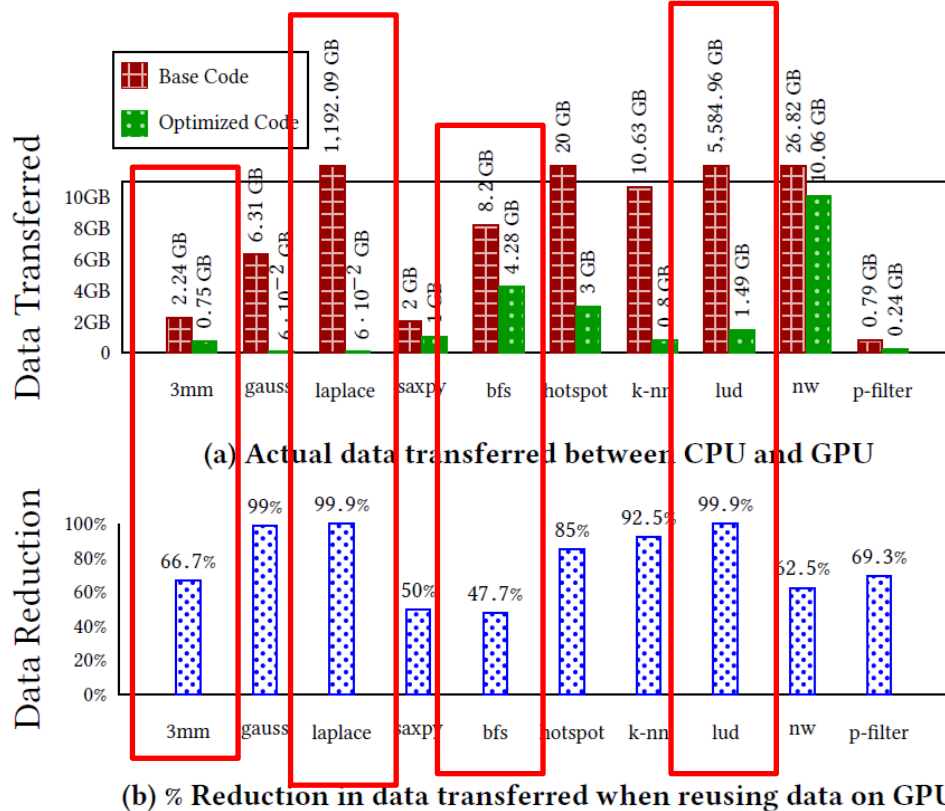
#pragma omp target data map(to:C) map(tofrom:D)
#pragma omp target teams distribute parallel for
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            for (int k = 0; k < N; k++)
                D[i][j] += temp[i][k] * C[k][j];
}
```

Experimental Setup

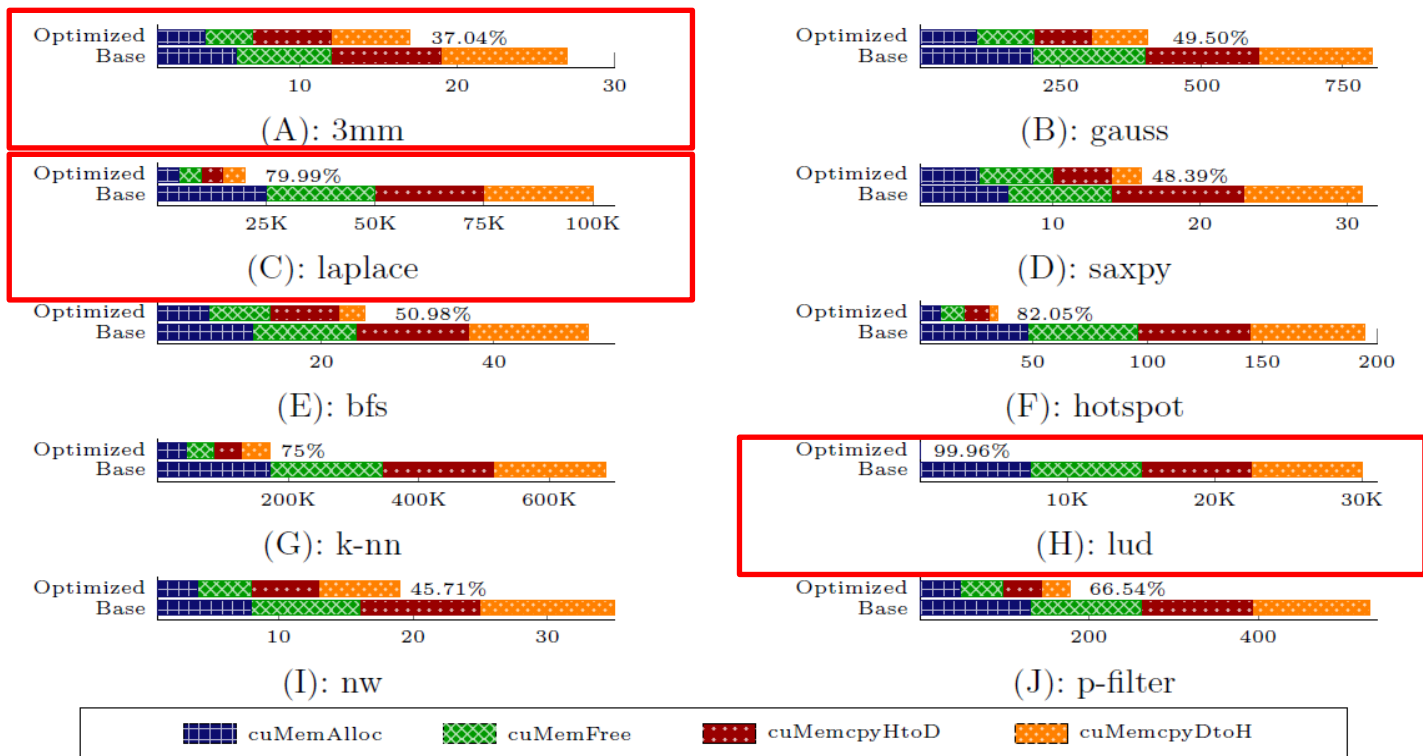


- ❖ SeaWulf cluster at Stony Brook University
- ❖ NVIDIA Tesla V100
- ❖ SOLLVE
 - LLVM version 8.0
- ❖ 4 microbenchmark and 6 application from Rodinia Benchmark Suite
 - Base Code
 - Optimized Code

RESULTS - REDUCTION



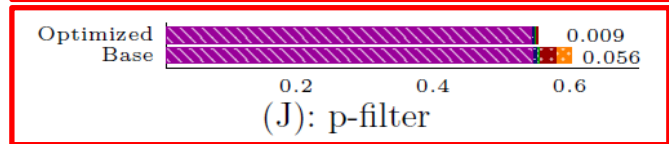
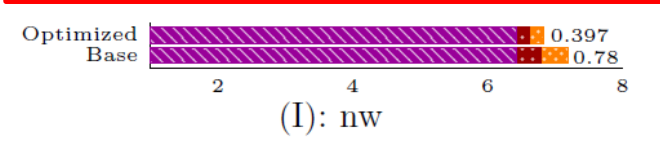
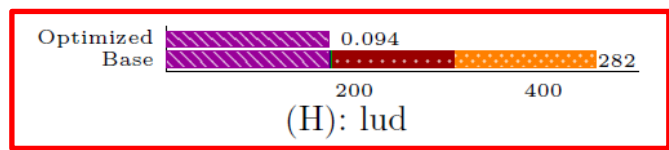
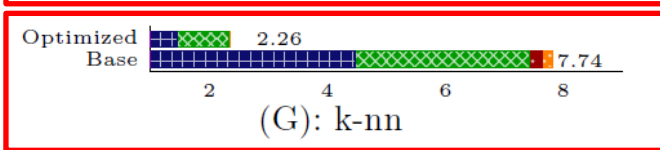
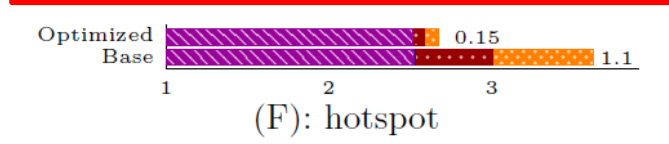
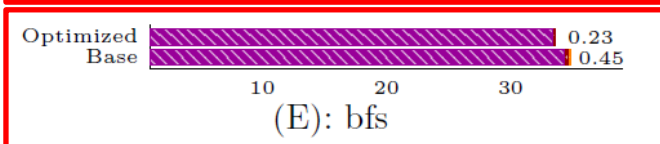
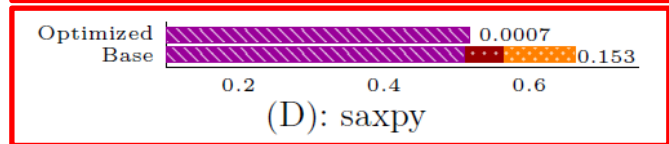
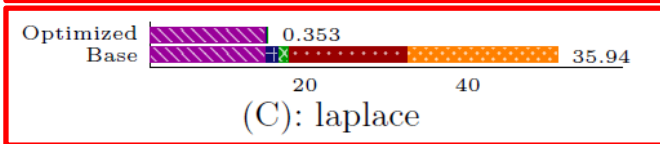
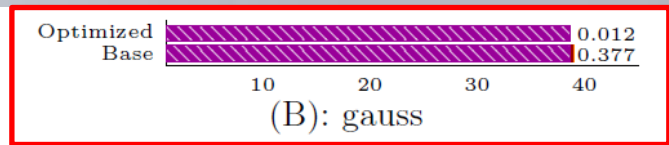
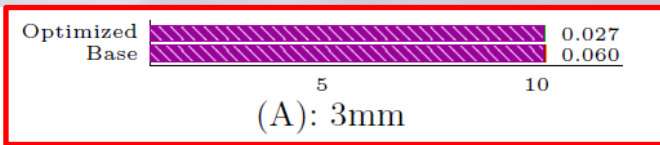
RESULTS – CUDA APIs



Number of calls to data transfer CUDA APIs.

The % at the tip of optimized code represents reduction in total number of calls when compared to base code.

RESULTS – EXECUTION TIME



Time taken (in sec) for different data management APIs and kernel computation time on V100 GPU.

The numbers at the tip of each graph represents the time taken for data transfer only (in sec).

Related Work



- ❖ Similar experiment ran using GCC with similar result
- ❖ Other works
 - *An asymmetric distributed shared memory model for heterogeneous parallel systems.* I Gelado, J E Stone, J Cabezas, S Patel, N Navarro, W M W Hwu
 - *Automatic cpu-gpu communication management and optimization.* T B Jablin, P Prabhu, J A Jablin, N P Johnson, S R Beard, D I August.
 - *OMPSAN: Static verification of OpenMP's data mapping constructs.* P Barua, J Shirako, W Tsang, J Paudel, W Chen, V Sarkar.

- ❖ If data is not reused on GPU
 - The performance of some application reduces significantly
- ❖ No loss of performance in any other cases
- ❖ User can accept or reject the transformation
- ❖ Future
 - Extend proximity check
 - Unified Memory
 - OpenMP 5.0
- ❖ Help other research aimed at Automatic GPU offloading of code

Conclusion & Future Work



THANK YOU!

Any questions?

You can find our work at:

- <https://github.com/almishra/data-reuse-analysis>

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

Special thanks to our colleague Dr. Chunhua Liao from Lawrence Livermore National Laboratory for his initial feedback and helpful discussions.

