

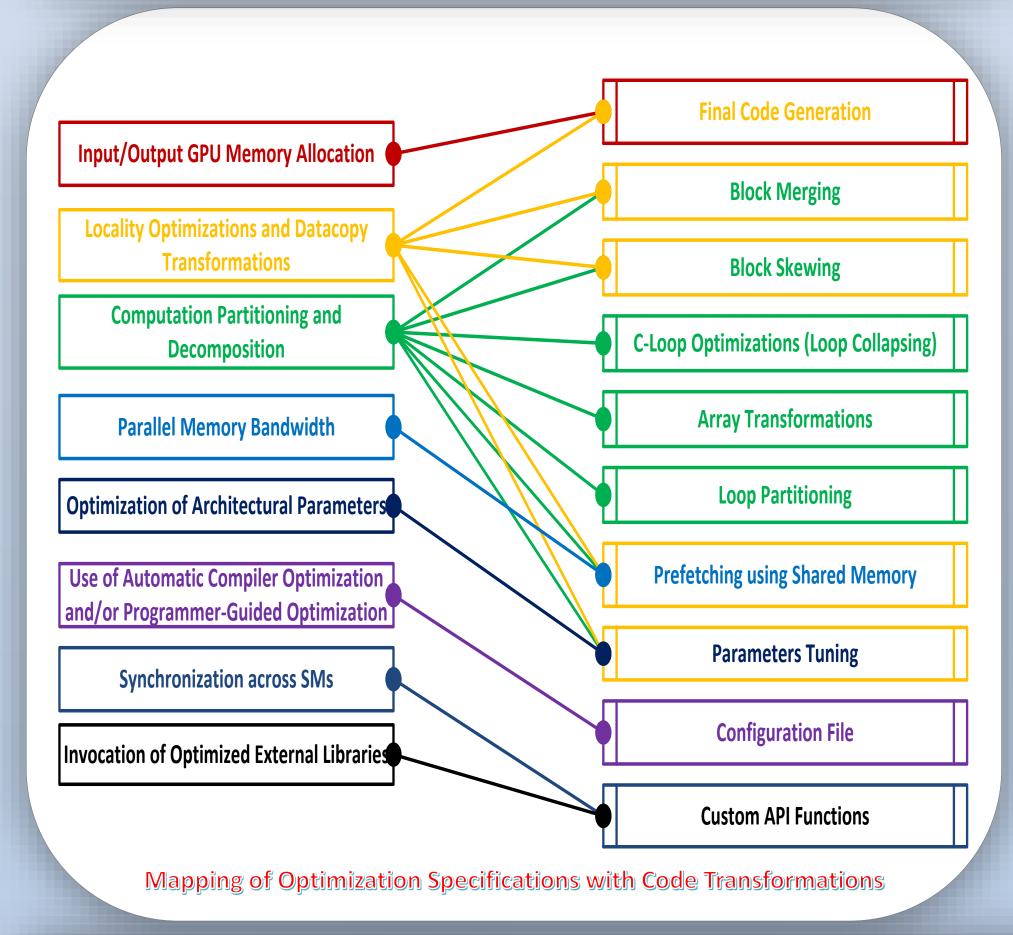
Optimization Specifications for CUDA Code Restructuring Tool

Ayaz. H. Khan. Computer Science Department, College of Computer, Qassim University, Email: ay.khan@qu.edu.sa.

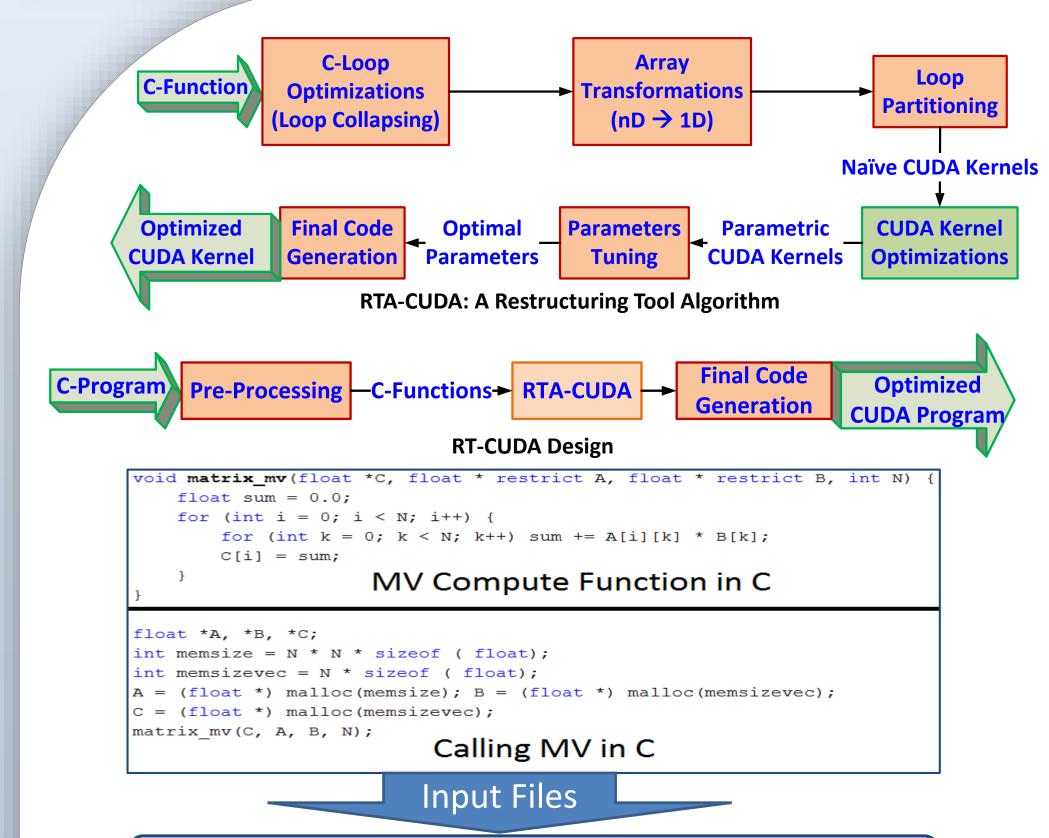
Abstract

In this work we have developed a restructuring software tool (RT-CUDA) following the proposed optimization specifications to bridge the gap between high-level languages and the machine dependent CUDA environment. RT-CUDA takes a C program and convert it into an optimized CUDA kernel with user directives in a configuration file for guiding the compiler. RT-CUDA also allows transparent invocation of the most optimized external math libraries like cuSparse and cuBLAS enabling efficient design of linear algebra solvers. We expect RT-CUDA to be needed by many KSA industries dealing with science and engineering simulation on massively parallel computers like NVIDIA GPUs.

Specifications



RT-CUDA Compiler and Its Application to Scientific Simulation

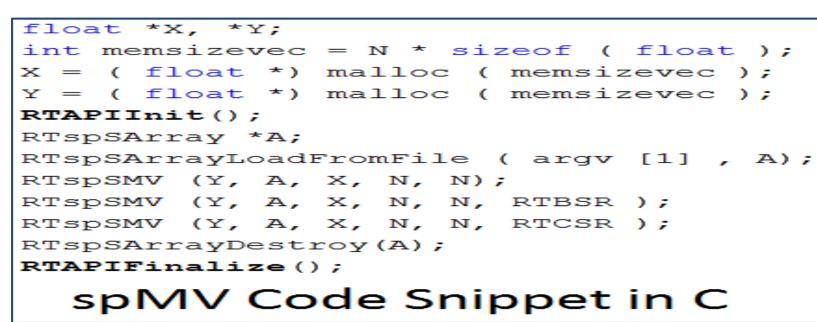


RT-CUDA Compiler

Output Files global__ void matrix_mv(float *C, float * __restrict__ A, float * __restrict__ B, int N) { shared float As[MERGE LEVEL][BLOCKSIZE]; float sum[MERGE LEVEL]; for (int i = 0; i < MERGE LEVEL; i++) sum [i] = 0.0; int tid = threadIdx .x; int bid = blockIdx .x; int i = (bid * BLOCKSIZE + tid) * MERGE LEVEL; for (int m = 0; m < MERGE LEVEL; m++) {</pre> As[m][tid] = A[((i + m)) * N + (k) + tid];__syncthreads(); for (k = 0; k < N - BLOCKSIZE; k += BLOCKSIZE) {</pre> for (int t = 0; t < BLOCKSIZE; t++) {</pre> float b = B[k + t];for (int m = 0; m < MERGE_LEVEL; m++) sum[m] += As[m][t] * b;</pre> __syncthreads(); for (int m = 0; m < MERGE_LEVEL; m++) {</pre> As[m][tid] = A[((i + m)) * N + (k + BLOCKSIZE) + tid];__syncthreads(); for (int t = 0; t < BLOCKSIZE; t++) {</pre> for (int m = 0; m < MERGE LEVEL; m++) sum[m] += As[m][t] * b;</pre> for (int m = 0; m < MERGE LEVEL; m++) C[(i + m)] = sum[m]; Generated MV CUDA Kernel dim3 grid(N / BLOCKSIZE / MERGE_LEVEL / SKEW_LEVEL, 1); matrix mv <<<grid, threads >>>(C, A, B, N); cudaDeviceSynchronize(); Calling MV in CUDA

Compiling Native C-code into optimized CUDA

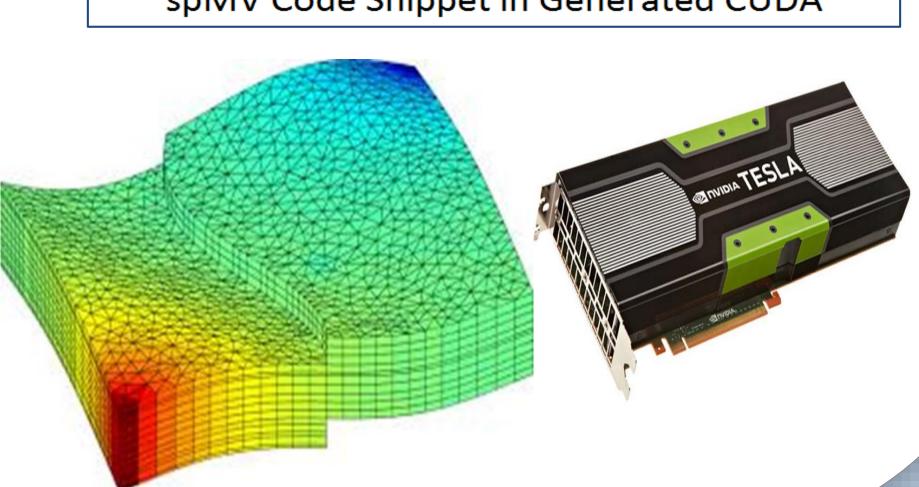
Naïve CUDA Block Merging Access in Loop Body CUDA Kernel Prefetching Using Shared Memory Place Prefetching Using Shared Memory Block Skewing CUDA Kernel CUDA Kernel Optimizations



Input Files

RT-CUDA Compiler

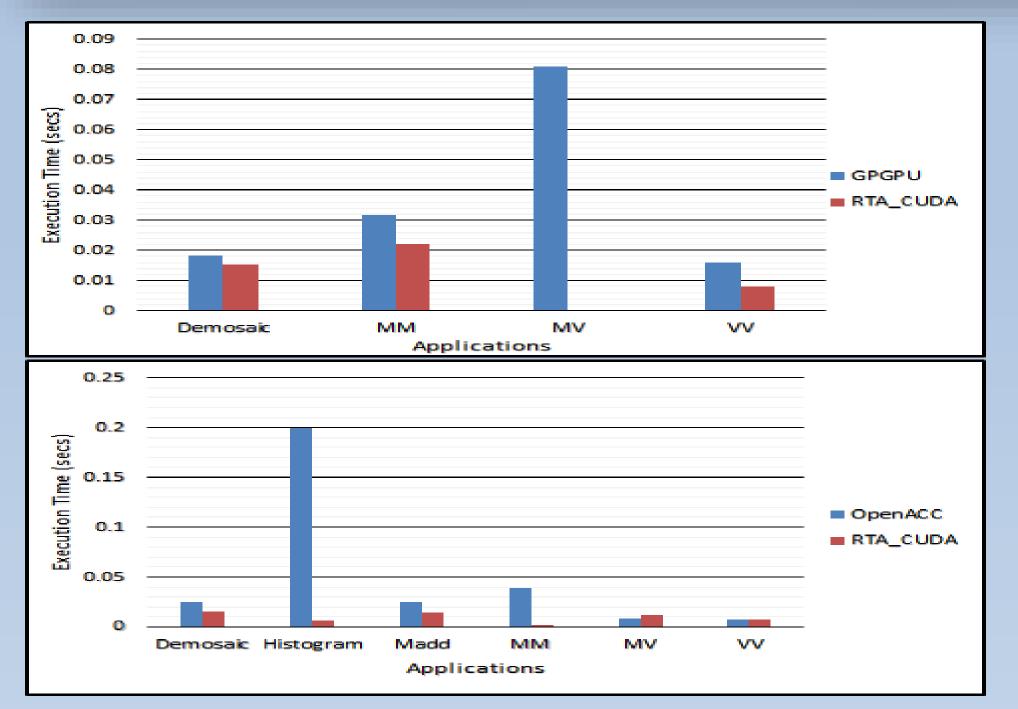
Output Files #include "rcuda.h" //includes definition of RT-CUDA API function float *X, *Y; int memsizevec = N * sizeof (float); **Compiling Calls** cudaMallocManaged (&X, memsizevec); cudaMallocManaged (&Y, memsizevec); to cuSPARSE RTAPIInit(); RTspSArray *A; and cuBLAS RTspSArrayLoadFromFile (argv [1] , A); **Optimized** RTspSMV (Y, A, X, N, N); RTspSMV (Y, A, X, N, N, RTBSR); Libraries RTspSMV (Y, A, X, N, N, RTCSR); RTspSArrayDestroy (A); RTAPIFinalize(); spMV Code Snippet in Generated CUDA

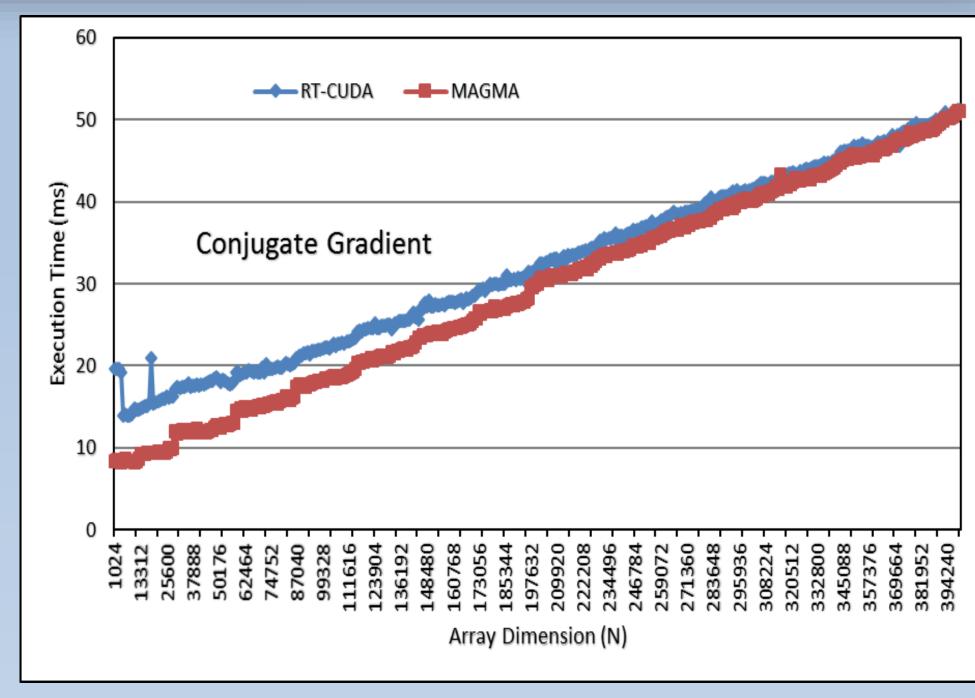


Ease and Efficient Library Coding of Large Scale Solvers

- RT-CUDA supports efficient development of sparse iterative linear solvers such as conjugate gradient to be used in reservoir simulation softwares
- RT-CUDA includes API functions to allocate and initialize sparse matrices with random sparsity as well as reading matrix from matrix market file
- RT-CUDA supports combination of user defined functions and invoking highly optimized library functions including cuBLAS and cuSparse library functions as shown in the example above
- RT-CUDA hides architectural details of the underlying GPU device that helps traditional C programmers to develop parallel programs in a fast and efficient manner

Benchmarking RT-CUDA using LAPACK, cuSparse, cuBLAS, MAGMA





Conclusions

- a) RT-CUDA a software compiler with best possible kernel optimizations to bridge the gap between high-level
- languages and the machine dependent CUDA and GPUsb) Obtained significant speedup over other compilers likeOpenACC and GPGPU compilers
- c) Enables transparent invocation of the most optimized external math libraries like cuSparse, and cuBLAS. For this, RT-CUDA uses interfacing APIs, error handing interpretation, and user transparent programming
- d) R-CUDA facilitates the design of efficient parallel software for developing parallel simulators (reservoir simulators, molecular dynamics, etc.) which are critical for Aramco and Oil and Gas industry in KSA
- e) RT-CUDA needed by many KSA industries dealing with science and engineering simulation on massively parallel computers like NVIDIA GPUs and Intel manycores.

