ARL-TN-0973 ● Sep 2019

DEVCOM
ARMY RESEARCH LABORATORY

Conversion of the Lagrangian Particle Dispersion Model (LPDM) Code to C Using Graphics Processing Unit (GPU) Computing

by Leelinda P Dawson and Yansen Wang

ARL-TN-0973 ● SEP 2019

Conversion of the Lagrangian Particle Dispersion Model (LPDM) Code to C Using Graphics Processing Unit (GPU) Computing

Leelinda P Dawson and Yansen Wang
Computational Information Sciences Directorate, CCDC Army Research Laboratory

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* September 2019	**2. REPORT TYPE** Technical Note	**3. DATES COVERED (From - To)** October 2018–September 2019
4. TITLE AND SUBTITLE Conversion of the Lagrangian Particle Dispersion Model (LPDM) Code to C Using Graphics Processing Unit (GPU) Computing		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Leelinda P Dawson and Yansen Wang		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) CCDC Army Research Laboratory ATTN: FCDD-RLC-EM 2800 Powder Mill Road, Adelphi, MD 20783-1138		**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TN-0973
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCID ID(s): Dawson, 0000-0003-4209-8459

14. ABSTRACT
The Lagrangian Particle Dispersion Model (LPDM) simulates the ensemble average transport of aerosols and gases in turbulent wind conditions, which can have a long execution time for it to be possibly useful to the Warfighter. Previously, some performance improvements were achieved by using two different approaches of graphics processing unit (GPU) computing technology with one using a Compute Unified Device Architecture (CUDA) Fortran interface via device code and the other one using CUDA host code only. However, it was determined later that converting the original LPDM code from Fortran to C programming language was needed to further increase its execution performance. This report documents the implementation approach of converting the original LPDM model code to C using GPU computing technology to improve its performance. The execution time of GPU-accelerated LPDM C application was faster than the original LPDM Fortran application without GPU computing.

15. SUBJECT TERMS
Lagrangian Particle Dispersion Model, LPDM, graphics processing unit, GPU, CUDA, C

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Leelinda P Dawson
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	UU	16	**19b. TELEPHONE NUMBER (Include area code)** 301-394-5636

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

1. Introduction

The Lagrangian Particle Dispersion Model (LPDM) simulates the ensemble average transport of aerosols and gases in turbulent wind conditions, which can have a long execution time for it to be possibly useful to the Warfighter on the battlefield. The LPDM can be coupled with the atmospheric boundary flow models[1,2] inline or offline for the transport and dispersion predictions. As discussed in Dawson[3] and Dawson and Wang,[4] there were some performance improvements of the LPDM using two techniques of graphics processing unit (GPU) computing technology with one using Compute Unified Device Architecture (CUDA) Fortran interface via device code and another one using CUDA host code only. However, additional performance improvement with the LPDM code was preferred. As a result, it was determined that converting the original LPDM Fortran code to C programming language[5] was required. It is assumed that this will help further maximize LPDM's execution performance since CUDA's programming framework was created in C.

The purpose of this document is to explain the implementation approach of converting the original LPDM Fortran model code to C programming language using GPU computing technology to improve its application performance. The execution time of GPU-accelerated LPDM C application was faster than the original LPDM Fortran application without GPU computing. However, there were some minor differences in the output of GPU-accelerated LPDM C model code that still need further investigation.

2. GPU and Conversion of the LPDM Code

The compute-intensive portions of a GPU-accelerated application are mainly offloaded to the GPU with many cores running in parallel and simultaneously, while the remainder of the application code runs on the central processing unit (CPU).[6] Thus, GPU-accelerated applications can run much faster than a CPU application. Similar to the approaches described in Dawson[3] and Dawson and Wang,[4] The US Army Combat Capabilities Development Command Army Research Laboratory's high-performance computer, Excalibur, was used for its GPU computing capability during the experiment.

The original LPDM (Wang et al.[7]) application code was developed in Fortran 90.[8] During the conversion of the LPDM Fortran code to C, CUDA 8.0[9] application interface for Compute Capability 3.5 was used, similar to the approach described in Dawson and Wang[4]. In contrast, OpenACC was not used and Nvidia's CUDA

Compiler (NVCC) was utilized to compile the LPDM's C model code during this experiment.

The original LPDM code contains over 500 lines of Fortran code. Each line was converted line-by-line to its C equivalent during the conversion process. There were some similarities with Fortran and C, but there were more notable differences with multidimensional arrays and loop ordering. The LPDM Fortran code contains over 10 three-dimensional arrays for various implementations, such as when using the random number generator for marking particle dispersions. Fortran multidimensional arrays are stored in memory with the most rapidly changing index coming first (e.g., A(i,j,k)), whereas C multidimensional arrays are stored in memory with the most rapidly changing index last (e.g., A[k][j][i]). In other words, each array was transposed during the conversion from Fortran to C, where Fortran is column-major order and C is row-major order (Fig. 1). Furthermore, the LPDM Fortran code contains approximately 20 do-while loops for various implementations. In the LPDM C code, all the do-while loops and one-dimensional array declarations with their assigned values were both converted to for loops. The Fortran arrays start at 1, while as C arrays start at 0. Therefore, each do-while loop and array declarations in Fortran started at 1, while each C loop and array declarations start at 0 (Fig. 1).

Original LPDM Fortran Code:

```
integer,parameter::no_pt=1000000               !number of particles
integer,parameter::nx=201,ny=201,nz=201        !number of grid points
real::aa(no_pt)                                !random number vectors
real::Tw_lag(nz)                         !lagrange integral time scale
real::sigma_w(nx,ny,nz)                    !Eulerian sigma w
integer::i2(no_pt),j2(no_pt),k2(no_pt)     !instantaneous i,j,k index for particles
real::dx,dy,dz                           !grid box dx,dy,dz in meters
real::w(no_pt),dw(no_pt)
real::x(no_pt),y(no_pt),z(no_pt)         !coordinate for each particle
…
k2(1:no_pt)=int(z(1:no_pt)/dz+1)
i2(1:no_pt)=int(x(1:no_pt)/dx+1)
j2(1:no_pt)=int(y(1:no_pt)/dy+1)
…
do ip=1,no_pt
if (out_flag(ip) .ne. 1 .and. ip .le. no_pt) then
dw(ip)= -(1./Tw_lag(k2(ip)))*w(ip)*dt  &
+sqrt(sigma_w(i2(ip),j2(ip),k2(ip))/Tw_lag(k2(ip)))*aa(ip)
w(ip)=w(ip)+dw(ip)
…
endif
enddo
…
```

LPDM C Code:

```
int const no_pt = 1000000;        // number of random elements/particles
int nx=201,ny=201,nz=201;         //number of grid points
float aa[no_pt];                  //random number vectors
float Tw_lag[nz];                 //lagrange integral time scale
float sigma_w[nz][ny][nx];        //Eulerian sigma w
int i2[no_pt],j2[no_pt],k2[no_pt];    //instantaneous i,j,k index for particles
float dx,dy,dz;                           //grid box dx,dy,dz in meters
float w[no_pt],dw[no_pt];
float x[no_pt],y[no_pt],z[no_pt];     //coordinate for each particle
…
for(i = 0; i < no_pt; ++i)
{
   k2[i] = (int)(z[i]/dz);
   i2[i] = (int)(x[i]/dx);
   j2[i] = (int)(y[i]/dy);
}
…
for(ip = 0; ip < no_pt; ++ip)
{
   if(out_flag[ip] != 1)
   {
      dw[ip] = -(1.0/Tw_lag[k2[ip]])*w[ip]*dt +
                sqrt(sigma_w[k2[ip]][j2[ip]][i2[ip]])/Tw_lag[k2[ip]]*aa[ip];
      w[ip] = w[ip] + dw[ip];
…
   }
}
…
```

Fig. 1 Sample of original LPDM Fortran code converted to C code

In addition to converting all the original LPDM code from Fortran to C, an initial effort was made to use GPU computing technology via CUDA to accelerate the LPDM C code. As discussed in Dawson,[3] the most intensive portion of the model computation in the original LPDM Fortran code is the Gaussian random process that simulates the diffusion by small turbulent eddies, and the function, *gaussdev*, was determined to be the hotspot for the LPDM code via the PGPROF profiler. CUDA has a built-in random number generator (RNG) library called CUDA Random Number Generation library (cuRAND) that produces high-performance GPU-accelerated random number generation from several RNGs distributions including Gaussian or normal distribution.[9] As a result, the LPDM Fortran code containing all instances of the *gaussdev* function was replaced with cuRAND host function calls in the LPDM C GPU-accelerated code, as shown in Fig. 2.

Original LPDM Fortran Code:

```
integer,parameter::no_pt=1000000
real::aa(no_pt)

do i=1,no_pt
  aa(i)=gaussdev()
enddo
```

LPDM GPU-Accelerated C Code:

```
int const no_pt = 1000000;
float aa[no_pt];
size_t const bytes = no_pt * sizeof(float);
float* inputD;
float* hostD;
float mean = 0.0, stdDev = 1.0;
curandGenerator_t gen;

hostD = (float*) malloc(bytes);
cudaMalloc((void**)&inputD, bytes);
cudaMemset(inputD,0,bytes);

curandCreateGenerator(&gen, CURAND_RNG_PSEUDO_XORWOW);
curandSetPseudoRandomGeneratorSeed(gen, clock());
curandGenerateNormal(gen, inputD, no_pt, mean, stdDev);
cudaMemcpy(hostD, inputD, bytes, cudaMemcpyDeviceToHost);
cudaDeviceSynchronize();
curandDestroyGenerator(gen);
```

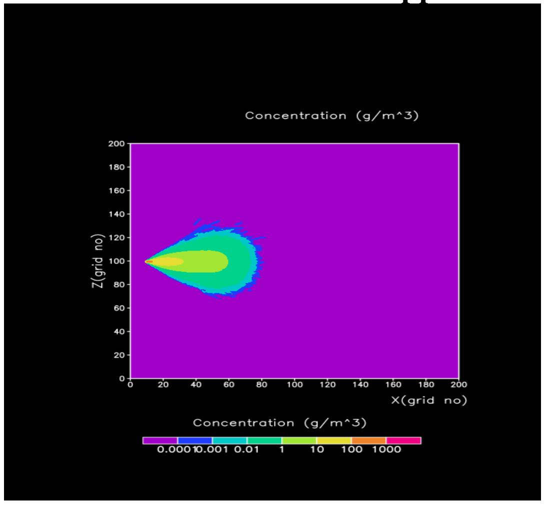
Fig. 2 Original LPDM Fortran code's performance hotspot and its equivalent GPU-accelerated C code

3. Results

As stated in Section 2, the original LPDM Fortran code was converted to a GPU-accelerated C application using line-by-line conversion and CUDA. The GPU C-accelerated application was faster than the original LPDM application without GPU computing. As discussed in Dawson,[3] the original LPDM Fortran application without GPU computing ran on the CPU with the execution time of 7 min and 57 s. On the other hand, the GPU-accelerated LPDM C application ran at 3 min and 29 s, which is approximately 2.28 times faster than the original LPDM application without GPU computing.

As shown in Fig. 3, the graphical data results simulate the average transport of aerosols and gases released under turbulent wind conditions using the original LPDM Fortran application without GPU computing versus the LPDM GPU-accelerated C application. The data results between the two applications have some minor differences, which will need to be further investigated as it relates to the main cause for them. One possible reason could be the differences in how the output binary files are created in Fortran versus C that needs be explored, especially since the output arrays are transposed, as discussed in Section 2. However, the effort in converting the LPDM Fortran code to C code using GPU computing was somewhat successful, since the LPDM execution time was reduced by the GPU-accelerated C application.

LPDM GPU-Accelerated C Application:



Original LPDM Fortran Application:

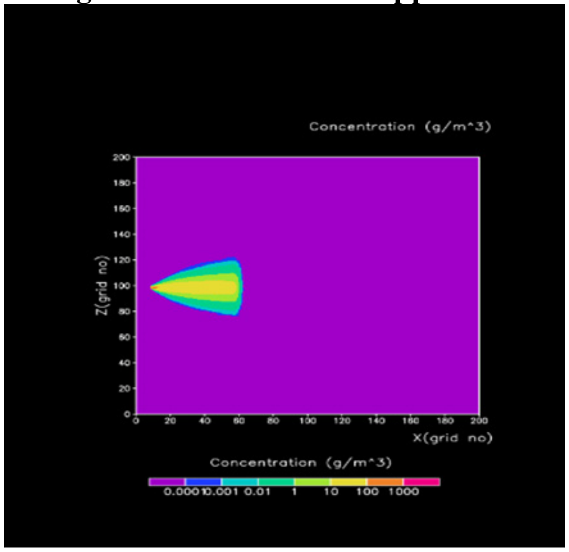


Fig. 3 Data results of original LPDM Fortran application and GPU-accelerated LPDM C application

4. Conclusion and Future Work

The original LPDM Fortran application was converted to a C GPU-accelerated application with some success since an improvement in the application's performance was achieved. The LPDM GPU-accelerated C application was over 2 times faster than the original LPDM Fortran application. However, additional work is needed to gain more performance with the LPDM C application.

The future work for the LPDM GPU-accelerated C application involves additional investigation in the cause for minor differences in the output from the original LPDM Fortran application. Once this is solved, the next step will be to start integrating various CUDA techniques to further optimize the LPDM C application. There are many CUDA programming techniques that can be explored, such as constant shared memory, asynchronous memory transfers with data streams, and dynamic parallelism. The goal is to experiment with these and other CUDA techniques to determine which technique and/or a combination of these techniques can be used to successfully maximize the LPDM application's performance. Then, the LPDM GPU-accelerated C application could possibly be used for rapid release and planning purposes on the battlefield.

5. References

1. Wang Y, Williamson C, Garvey D, Chang S, Cogan J. Application of a multigrid method to a mass consistent diagnostic wind model. J Appl Meteorology. 2005;44:1078–1089.

2. Wang Y, MacCall BT, Hocut CM, Zeng X, Fernando HJS. Simulation of stratified flows over ridge using a lattice Boltzmann model. Environ Fluid Mech. 2018. https://doi.org/10.1007/s10652-018-9599-3.

3. Dawson L. The performance improvement of the Lagrangian Particle Dispersion Model (LPDM) using graphics processing unit (GPU) computing. Adelphi (MD): Army Research Laboratory (US); 2017 Aug. Report No.: ARL-TR-8110.

4. Dawson L, Wang Y. The second approach to improving the performance of the Lagrangian Particle Dispersion Model (LPDM) using graphics processing unit (GPU) computing. Adelphi (MD): Army Research Laboratory (US); 2018 Sep. Report No.: ARL-TN-0907.

5. C language. [accessed 2019 Sep]. https://en.cppreference.com/w/c/language.

6. Krewell K. What is the difference between a CPU and a GPU? Santa Clara (CA): Nvidia Corporation; 2009 [accessed 2019 Sep]. https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/.

7. Wang Y, Miller D, Anderson D, McManus. A Lagrangian stochastic model for aerial spray transport above an oak forest. Agricultural and Forest Meteorology. 1995;76:277–291. ISSN 0168-1923.

8. Fortran90. Fortran90 1.0 documentation: 2018 [accessed 2019 Sep]. http://www.fortran90.org/.

9. cuRAND. Santa Clara (CA): Nvidia Corporation; 2017 [accessed 2018 Aug]. https://developer.nvidia.com/curand.

List of Symbols, Abbreviations, and Acronyms

CPU	central processing unit
CUDA	Compute Unified Device Architecture
cuRAND	CUDA Random Number Generation Library
GPU	graphics processing unit
LPDM	Lagrangian Particle Dispersion Model
NVCC	Nvidia’s CUDA Compiler
RNG	random number generator

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	CCDC ARL FCDD RLD CL TECH LIB
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
4 (PDF)	CCDC ARL FCDD RLC E B MACCALL FCDD RLC EM L DAWSON Y WANG FCDD RLC NB E CHIN