Meso-Scale Radioactive Dispersion Modelling using GPU

Sunarko¹*, Zaki Suud²

¹BATAN, Jl. Kuningan Barat, Jakarta 10270, Indonesia
²ITB, Jl. Ganesha 10, Bandung 50254, Indonesia

*Email: sunarko@batan.go.id

Abstract. Lagrangian Particle Dispersion Method (LPDM) is applied to model atmospheric dispersion of radioactive material in a meso-scale of a few tens of kilometers for site study purpose. Empirical relationships are used to determine the dispersion coefficient for various atmospheric stabilities. Diagnostic 3-D wind field is created based on data from a meteorological station using mass-conservation principle. Particles imitating radioactive pollutant are dispersed in the wind-field as a point source. Time-integrated air concentration is calculated using kernel density estimator (KDE) in the lowest layer of the atmosphere. Parallel code is developed for GTX-660Ti GPU with a total of 1344 scalar processors using CUDA programming. Significant speedup of about 20 times is achieved compared to the serial version of the code while accuracy is kept at reasonable level. Only small differences in particle positions and grid doses are observed when using the same sets of random number and meteorological data in both CPU and GPU versions of the code.

1. Introduction

Lagrangian Particle Dispersion Method (LPDM) is now commonly used in atmospheric dispersion studies. This method uses hypothetical particles to represent substances undergo advection by wind and diffusion process in the atmosphere as well as deposition to the ground. At any time, these particles will reside in a certain layer of the atmosphere and concentration calculation should be made for father analysis. Kernel Density Estimator (KDE) is one of many methods that can be employed to provide estimation for the concentration. In application where time-integrated concentration is required such as for dose calculation in the dispersion of radioactive substances, running time becomes crucial because concentration calculation should be performed over a specified interval.

The use of GPU arisen from the demand for real-time, high-definition 3D graphics has driven the development of programmable Graphic Processor Unit or GPU into a highly parallel, multithread, many-core processor with tremendous computational power and high memory bandwidth. In the past decade, parallel computation using Graphical Processing Unit (GPU) has gained broader acceptance in scientific computation including dispersion modeling. The use of GPU has grown exponentially due to the appearance of new architectures and programming models such as NVIDIA CUDA (formerly abbreviated from Compute Unified Device Architecture)[7]. Compared to a multicore CPU, a GPU can contain much larger number of processors and can provide speedups of more than 100 times in science and engineering applications[3]. This advantage can be utilized in KDE calculation since it is
inherently parallel. Computation in a grid point can be done independent of other grid points each with the same set of particles.

In this paper we present a parallel code for radioactive dose calculation from atmospheric dispersion of radionuclides using a GPU and measure its performance improvement over a single-threaded code for CPU application.

2. Model description

There are several main processes in the model necessary for facilitating particle dispersion such as wind-field preparation, elevation model, dispersion coefficient, concentration calculation method and random number generation. The flow of process in the code is provided in Figure 1.

Lagrangian Particle Dispersion Method (LPDM) is run by tracking a number of particles in a flow field. LPDM uses stochastic differential equation to explain the similar process as the advection-diffusion equation in Lagrangian framework. Stochastic differential equation for the movement of ideal fluid in three dimensions is:

\[
\begin{align*}
    dx &= \bar{u}dt + (2K_x)^{1/2}dW_x \\
    dy &= \bar{v}dt + (2K_y)^{1/2}dW_y \\
    dz &= \bar{w}dt + \frac{\partial K_z}{\partial z} dt + (2K_z)^{1/2}dW_z
\end{align*}
\]

where \(dW_{x,y,z}\) is a random number with zero average and variance \(dt\), namely \(d\bar{W} = 0\) and \(d\bar{W}^2 = dt\). In equation (4) and (5) it is assumed that turbulence is homogeneous in \(x\) and \(y\) directions. The above equation can be integrated with time to obtain particle path which represent the movement of each individual particle. In numerical calculation of dispersion, the number of released particle is large, each with its own label and certain mass or activity. Concentration at time \(t\) can be calculated from locations of particles at time \(t\) and particle activity.

Implementation of random-walk method is explained for vertical direction in Error! Reference source not found.. The equation for horizontal direction has the same form with the vertical one but without the differential form of eddy diffusivity \((K)\). The vertical and horizontal dispersion coefficient is taken from Caughey et al. [11] for various stability conditions.

Three-dimensional meteorological data is not always available from meteorological agencies. Frequently, site-specific meteorological data should be obtained by way of one or more monitoring stations with limited height and parameters. Data is usually recorded over a certain period of time,  

\[\text{Figure 1. Process flow}^1\]

---

1. Figure adapted from: https://str.llnl.gov/str/Baskett.html
often hourly. Consequently, the 3-dimensional wind-field should be prepared hourly for the whole duration of the simulation. Three dimensional wind field is prepared using mass-conservation principle such as the Mass Consistent Model (MCM) [2] which is sufficient for modeling up to the meso-scale level. This method is capable of utilizing data from a limited data from a meteorological tower while at the same time provide usable data for particle advection. The wind field is produced in the CPU and then transferred to the 3D texture memory in the device before being used by the kernel. MCM was proposed by Sasaki in 1958 and implemented by Sherman in 1978. MCM employs minimization of functional:

\[
E(u, v, w, \lambda) = \int \left[ \alpha_1^2(u - u_0)^2 + \alpha_2^2(v - v_0)^2 + \lambda \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \right] dx \, dy \, dz
\]

Equation (4) is solved using finite-difference method in a 101x101x31 rectangular grid to obtain 3-dimensional wind field. The spacings of the grid are \( \Delta x = 1 \, km, \Delta y = 1km, \Delta z = 50m \). Air is assumed as incompressible and meteorological data from one station is used (3-dimensional wind field). The spacings of the grid are determined by the atmospheric stability category. The lagrange multiplier \( \lambda \) is used to define the boundary condition.

After a number of particles undergo transport and diffusion process and their locations are noted, the concentration at a certain location and time can be determined by counting the number of particles on a sampling area centered at the grid point \((x, y, z)\) known as box-counting method as follows:

\[
C(x, y) = \frac{1}{\Delta x \, \Delta y \, \Delta z} \sum_{i=1}^{N} m_{pi} l
\]

\[
l = 1 \quad \text{if} \quad |X_i - x| < \Delta x/2, \quad |Y_i - y| < \Delta y/2, \quad |Z_i - z| < \Delta z/2 \text{ others}
\]

\[
m_{pi}, X_i, Y_i, Z_i \text{ are mass or activity in the coordinate } (X, Y, Z) \text{ of particle } i. \text{ Concentration is calculated in an area } \Delta x\Delta y\Delta z \text{ with } N \text{ number of particles. The increase in } N \text{ will increase computation time and an increase in sampling area will decrease the resolution.}
\]

Computation efficiency can be increased using KDE. Error! Reference source not found.. Concentration on a grid point can be calculated from the sum of contribution from each particle:

\[
C(x, y) = \sum_{i=1}^{N} \frac{m_{pi}}{h_{xi} h_{yi} h_{zi}} K \left( r_{xi, r_{yi, r_{zi}} \right)
\]

\[
(1/h_{xi} h_{yi} h_{zi}) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} K dx \, dy \, dz = 1
\]

\[
r_{xi} = (X_i - x)/h_{xi}, \quad r_{yi} = (Y_i - y)/h_{yi}, \quad r_{zi} = (Z_i - z)/h_{zi} \text{ are distances of particle } (X_i, Y_i, Z_i) \text{ with grid (x, y, z). Parameters } h_x, h_y, \text{ and } h_z \text{ are bandwidths to determine the degree of smoothness on each coordinate. Average concentration can be calculated using a simple uniform kernel as follows:}
\]

\[
K(r_{x}, r_{y}, r_{z}) = \frac{1}{8} l_x l_y l_z
\]

\[
l_{x,y,z} = \begin{cases} 1 & r_{x,y,z}^2 < 1 \\ 0 & \text{ others} \end{cases}
\]
Bandwidth is proportional to the grid space used: $h_x = \alpha \Delta x$, $h_y = \alpha \Delta y$, $h_z = \alpha \Delta z$. $\alpha = 1.0$ is used to obtain smooth concentration profile. A particle can contribute to the concentration on several grid points at a time.

3. Methodology
MODAL will be used to calculate external gamma exposure from I-131, Cs-137, Kr-88 and Xe-133 isotopes emitted through a 50 meter high stack in 6-hour duration. The modeled area is 100 km $\times$ 100 km with Cartesian grid size of $\Delta x = 1$ km, $\Delta y = 1$ km and $\Delta z = 0.05$ km. Total height of the model is 1500 meters. The air is incompressible and has uniform density anywhere in the grid.

Simulations are performed using isotopes of Cs-137, I-131, Kr-88, and Xe-133 each with a total strength of 0.011, 15.04, 62.000, and 8.37 million Curies respectively. The plume is released from a 50 m high stack at a constant rate from second 8.500 to 30.000 after the actual accident happens. Two scenarios are prepared each containing different amount of Xe-133 release provided in Table 1. For each isotope, maximum value is taken from available data for conservative assumption.

A number of simulations are performed representing dry and rain condition with 5, 10, 15 and 20 mm/hr rain intensities. Worst-case meteorological condition is picked from meteorological database from on-site monitoring program performed during site feasibility study in 1994-1995 periods. Worst-case condition in this paper is defined as condition where wind is blowing to a consistent sector during the entire duration of radioactive release. This type of atmospheric condition will tend to produce higher TIAC for locations in the downwind direction from the source emission. Frequency analysis is performed to pick a period in the database where wind blow consistency is found.

Parallel application
The CUDA programming model allows the launching of hundreds to thousands of GPU threads. CUDA comprises a programming language accompanied by the necessary libraries to support the execution of code on NVIDIA GPUs. A program that uses CUDA must be structured on kernels. These kernels are destined to run on the GPU (device) while the main routine is running on the CPU (host). In order to initiate a kernel execution using CUDA, one must define a mapping of application threads into blocks and accordingly a mapping of such blocks on a grid. Device code is executed by many independent threads, which are hierarchically organized; groups of threads are organized into grid. Within a block, 32 threads are organized into a group referred to as warp. Each streaming multiprocessor (MP) schedules and executes threads at the warp level[8]. A warp is the minimum scheduling unit. The device executes an instruction for all threads within a warp before launching the next instruction. Threads within a warp physically execute the same instruction in each cycle. The best performance is achieved when the number of threads in a thread-block is a multiple of the warp size[7].

CPU and GPU have their own separated memory and therefore data that is used by the GPU should be transferred first to the GPU before calculation can take place. The result of the calculation at the end of the process should be copied back to the CPU after the process in GPU is finished.

All compute devices follow the IEEE 754-2008 standard for binary floating-point arithmetic with a number of deviations. The existing discrepancy in floating-point capability between CPU and GPU is because GPU is specialized for compute-intensive, highly parallel computation which devotes more transistors to data processing rather than data caching and flow control[4]. It is important that this discrepancy does not cause large deviation in the calculation result between CPU and GPU systems.

Implementation
Wind-field, elevation model, dispersion coefficient and random number generations are performed by the CPU using mass-consistent method, Shepherd’s method, empirical relationships and Marsenne-Twister algorithm respectively. Particle dispersion, concentration calculation and integration processes are performed by the GPU due to the large number of particles and grid coordinates being used.
The particles are released in plume segments or puffs with duration of a number of seconds and therefore one hour of simulation may be comprised of several segments. One segment is propagated based on the particle locations of previous segment to save computation time. This way the texture cache memory can be fully utilized because particle positions in a puff are more localized in space. Each particle carries position, lifetime and status information. The activity is proportional to the total release of the hour divided by the number of particle released during the hour and undergoing radioactive decay as the time progresses. At the end of each plume travel, concentrations are taken and time-integrated concentration is obtained for each grid point. Concentrations are calculated using kernel density estimator for the lowest layer of the atmosphere where effect to human health is the most significant.

The dispersion coefficients \( (K_x, K_y) \) are calculated in the GPU to avoid errors from interpolation when 1D texture memory is used to store CPU generated tables. Random numbers used in the dispersion part of the simulation is generated in the CPU for every plume step and will be copied to the GPU prior to usage. The routine is performed using Marsenne-Twister library in C[10] with a period of \( 2^{19937} \).

Parallel processing is performed for particle dispersion and concentration calculation. In particle dispersion process, advection vector and dispersion coefficients are read from the 3D and 1D textures respectively using current particle position. Each stream processor is responsible for one particle at a time and there are no interactions among particles. Particle positions, random numbers, dry and wet deposition coefficients are first copied in coalescent from general memory to shared memory in BLOCKSIZE chunks for faster processing. The process is repeated as necessary until all particles are processed for the current puff step. New particle positions are stored back to the general memory for concentration calculation.

In the concentration calculation, each stream processor is assigned to a grid coordinate and calculation is done using the position of particles produced in the earlier process. Due to the limited capacity of shared memory, particles are copied to shared memory and used in the calculation in stages each of BLOCKSIZE chunks. These particles are shared by the grid coordinates in the same block in the calculation.

4. Results and discussion

Comparison of calculation result between CPU and GPU is performed by comparing the distance between particles. In this case, a same set of random numbers is used in both version of the code. For every hour, 18000 particles are released in groups of puff. The duration of each puff is 60 seconds and therefore there are 60 puffs in one hour, each containing 300 particles which are further organized into 6 smaller groups containing 50 particles. The first group released will travel the furthest and will likely be further separated than subsequent groups. The distance between particles in each group is calculated and averaged and their values are provided in Figure 4. Simulated wind condition containing fluctuation of wind data is used. Minor numerical error arises from the differences in mathematical functions differences between CPU and GPU[1]. However, this small error may translate into larger numerical error because of the difference in wind vectors being used in the subsequent steps. The more complex the wind pattern, the larger is the expected deviation.
Figure 2. Particle separation for each particle group.

The effect of particle separation will be carried forward to the concentration at grid points. Concentrations are calculated using Kernel Density Estimation for the lowest 5 meter layer because it is the most important layer in dispersion calculation. Calculations are performed for 101 x 101 grid points and the result of non-zero grid dose is provided in Figure 4. The above and below lines are plus and minus one standard deviation respectively.

Using a linear fit, it is found that the dose resulting for GPU calculation is about 15.6% higher. This is because more particles are dispersed from emission height into the lowest layer of the atmosphere where concentration calculations are performed. Dispersion and TIAC are calculated by the GPU involving trilinear interpolation of wind field, dispersion coefficient and concentration calculations. Only random number generation and wind field calculation are performed by the CPU.

Performance.

Simulations with different number of particles are also performed on a 3.0 GHz Core i5-3330 with 2 GB of RAM and GPU GTX660Ti with 1344 scalar processor and 2 GB memory. Figure 5 shows the speedup gain using the GPU. Speedup is contributed by many factors such as memory bandwidth, faster access to constant and shared memory as well as memory caching in GPU. Acceleration is also
provided by the readily available trilinear interpolation function in CUDA when 3D texture memory is used [1].

![Figure 4. Speedup by GPU application.](image1.png)

The speedup achieved is increasing with the increasing number of particles in use. In this experiment, approximately linear growth in speedup can be identified after 28 800 particles or more are used. CPU calculation was stopped at 144 000 particles/hour because of limited memory capacity.

![Figure 5. Running time.](image2.png)

Molnar measure performance without performing concentration integration. In this paper, concentration integration is performed every 60 second to calculate the time-integrated air concentration (TIAC) at each grid point which directly represents the dose received. Due to the limitation in shared memory capacity, the random number used in dispersion calculation is being read directly from general memory which slows down running time.

Zeng et al. [9] has also performed a parallelization based on another Lagrangian particle dispersion method named FLEXPART using 2.0 GHz and 4 GB CPU and comparing the performance against Tesla C1060 1.3 GHz and 4 GB memory GPU and achieved an acceleration of about 10 times.
5. Conclusion
GPU has been found to be a cost-effective solution in accelerating computational power for LPDM. Parallel version LPDM has achieved multiplication factor of about 20 and will increase with the increasing number of particles used. Differences in computation method between CPU and GPU systems as indicated by separation of particles at the end of 1 hour simulation may reach as far as 4 meters when the same set of random numbers and wind-field are used. The doses at grid points are slightly higher for computation using GPU by about 15.6% as a result of more particle dispersion into concentration calculation layer.

Acknowledgment
The authors acknowledge the Ministry of Research and Technology and Nuclear Physics Laboratory of Institute Teknologi Bandung for supporting this research.

References
[1] Molnar Jr. F, Szakaly T, Meszaros R and Lagzi L 2010 Computer Physics Communication 181 105.
[2] Sherman C A 1977 Journal of Applied Meteorology 17 312.
[3] NVIDIA Corporation 2010 NVIDIA CUDA programming Guide Version 2.3.1.
[4] NVIDIA Corporation 2010 NVIDIA CUDA C programming Guide Version 4.2.
[5] Uliasz M 1994 Lagrangian Particle Dispersion Modeling in Mesoscale Applications. Environmental Modelling II. ed. P. Zannetti (Southampton, Boston: Computational Mechanics Publication p 71)
[6] Sunarko 2009 Model dispersi partikel untuk perhitungan lepasan radioaktif (Studi kasus semenanjung Muria-Jawa Tengah) (Institut Teknologi Bandung : Master-Thesis).
[7] Jimenez J and Ruiz de Miras J 2012 Computer Methods and Programs in Biomedicine 108 1229.
[8] Kalantzis G and Tachibana H 2014 Computer Methods and Programs in Biomedicine 113 116.
[9] Zeng J, Matsunaga T and Mukai H 2010 Using NVIDIA GPU for Modelling the Lagrangian Particle Dispersion in the Atmosphere, International Environmental Modelling and Software Society (iEMSs) (Ottawa, Canada: International Congress on Environmental Modelling and Software Modelling for Environment’s Sake, Fifth Biennial Meeting).
[10] Matsumoto M and Nishimura T http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html.
[11] Caughey S J, Wyngaard J C and Kaimal J C 1979 J. Atmos. Sci. 36 1041.