DEVELOPMENT OF A NEW METHODOLOGY FOR IMPROVING URBAN FAST RESPONSE LAGRANGIAN DISPERSION SIMULATION VIA PARALLELISM ON THE GRAPHICS PROCESSING UNIT

P. Willemsen[1], A. Norgren[1], B. Singh[2] and E.R. Pardyjak[2]
[1]Department of Computer Science, University of Minnesota Duluth, USA
[2]Department of Mechanical Engineering, University of Utah, Salt Lake City, USA

INTRODUCTION

Recent trends in computing have shifted toward multi-core processors and programmable graphics processors with highly parallel data paths for processing geometry and pixels. Multi-core machines are now readily available with 2 cores, but machines with 4, 8, and even 16 cores are projected for the near future. Data parallelism in modern graphics cards is also increasing with raw performance of graphics processing units (GPUs) surpassing performance of CPUs. While initially specialized for processing computer graphics, GPUs can be programmed for general-purpose computations using high level programming languages similar to C. As a result, GPUs have become useful computational tools providing highly parallel data paths to accelerate a wide range of scientific and simulation applications.

One area of simulation that could greatly benefit from inexpensive parallelization is emergency response transport and dispersion modelling in urban areas. For the present work, we have attempted a GPU implementation of a simple Lagrangian dispersion model based on the Quick Urban and Industrial Complex (QUIC) Dispersion Modelling System (*Pardyjak, E.R. and M.J. Brown*, 2001;*Williams et al.*, 2002; *Nelson et al.*, 2006). The primary objectives of this work have been: (i) to implement a random walk particle model on the GPU, (ii) to validate the model against the standard CPU implementation and existing analytical solutions, (iii) to investigate issues and challenges associated with this unique implementation, and (iv) to understand the performance gains that can be realized through such an implementation. An additional immediate benefit of this approach is the real-time visualization of particle dispersion. Because the domain data are already loaded into the GPU memory, rendering these data to the screen after each simulation step is a fairly trivial process and affords a convenient means for visualizing the dispersion field. This is particularly important for virtual environment applications where a physical system must be integrated with a dispersion model running in near real-time for applications such as emergency response training exercises. In fact, the authors and colleagues are using the methodology developed here to model environmental flows in virtual environments (e.g., *Hollerbach, J. et al. 2005.*; *Kirkman, R., et al. 2006; Kulkarni, S.D et al. 2007*).

Much work has been focused over the last couple of years on utilizing the GPU to solve problems not associated with computer graphics or rendering. The GPU, and in particular, the pixel processing (or rasterization) component on the GPU, is a highly parallel stream processor capable of floating point computations (currently only 32-bit precision). The pixel processors (also referred to as fragment processors) are ideal for memory bound and compute bound applications in which the computations can be transformed into a SIMD (single-instruction, multiple-data) stream computation to be executed on the graphics hardware. On current hardware, it is possible for 128 fragments to be processed simultaneously. Examples of using the GPU are numerous in the literature and include computing and applying FFT to images (*Moreland & Angel, 2003*), numerically solving the Navier-Stokes equations (*Scheidegger, C. et al., 2005*), solving multigrid problems (*Bolz, J. et al., 2003; Goodnight, N.*

et al, 2003), solving dense linear systems (*Galoppo, N. et al., 2005*), and cloud dynamics (*Harris, M.J. et al., 2003*). Simple particle simulations involving one million particles have been run at interactive frame rates using the GPU (*Kipfer, P. et al., 2004*). The common point of these examples is that algorithm performance on the GPU can outperform the equivalent CPU computations. However, much effort generally goes into keeping data and computation on the graphics hardware (and hence off of the CPU) to maximize performance. Our efforts build on these techniques and work towards creating a viable solution for real-time simulation and visualization of atmospheric flows.

GPU Simulation Methodology

Computing particle dispersion on the GPU requires that the simulation be constructed to fit within the constraints of the GPU architecture to take advantage of the highly parallel pixel processing elements on the GPU. This requires that the computations for performing advection or any other per-particle processing be coded using a specialized shading programming language and that all data representing particle positions, wind fields, or other quantities be transformed into 2D textures. Textures are the primary memory structure on the GPU normally used to represent 2D images for use in texturing geometry in video games. However, in the context of general-purpose computation, textures become data sources. Our primary data source is a 2D array of particle positions. This 2D array represents all of the particles active within the simulation and is the data that the pixel processors operate on to perform an advection step. We achieve data-parallelism by programming the pixel processors to perform particle advection on this 2D array of particle positions. Since there are upwards of 128 pixel processors, hundreds of particles are operated on simultaneously. After completion of one simulation time step, *all* particles in the 2D array will have been advected. Rather than iterating over all the particles individually as is the case with the CPU implementation of QUIC, we supply a single advection operation by way of a small set of pixel programs (called shader programs) and tell the pixel processors to apply that operation to all of the particles in the 2D particle array as part of the normal screen refresh on the monitor. After the particles have been advected, we are immediately able to visualize them and show the results to the user. Note that simulations without visualization increases performance.

Additional 3D data sources, such as the wind field, are transformed from 3D to a 2D texture by splaying out the vertical slices of the wind field onto a single plane. To locate the wind field velocity at or near a particle position, we use GPU texture lookup functions to lookup the velocity in the wind field texture using the particle's position as the index. Similar transformations and operations are performed to obtain the fluctuating wind field quantities. Our application has been programmed using the OpenGL Shading Language and runs on Linux, OS X, and Windows machines equipped with a modern 3D graphics card, such as those made by NVIDIA or ATI. The results reported below were run on a 2.4 Ghz Intel Core 2 Duo Processor with an NVIDIA GeForce 8800 video card.

DESCRIPTION OF THE TEST CASE

The basic validation strategy used here is to compare the GPU implementation of QUIC-Plume to the CPU implementation and existing analytical solutions for a simple test case. For validation, an idealized continuous point source release (i.e., steady state, horizontally homogeneous, neutral atmospheric stability, constant wind speed, and constant eddy diffusivity) is compared to the classical Gaussian solution. This was previously done by *Singh, B. et al.* (2004) for the initial validation of the standard CPU implementation of QUIC-Plume. For additional details of the test case see *Singh, B. et al.* (2004). For the idealized case

considered here, the QUIC-Plume model is significantly simplified. The basic equations for the model are shown below in Eqs. 1-5 (following *Rodean, H.C.*, 1996) where the flow has been decomposed into mean and fluctuating quantities (i.e., $u_i = U_i + u'_i$).

$$x_i = x_{p,i} + U_i \Delta t + \frac{u'_{p,i} + u'_i}{2} \Delta t \quad (1)$$

$$u'_i = u'_{p,i} + du'_i \quad (2)$$

$$du'_1 = -\frac{C_o e}{2}\left(l_{11}u'_{p,1} + l_{13}u'_{p,3}\right)dt + \left(C_o e dt\right)^{1/2} dx_1 \quad (3)$$

$$du'_2 = -\frac{C_o e}{2}\left(l_{22}u'_{p,2}\right)dt + \left(C_o e dt\right)^{1/2} dx_2 \quad (4)$$

$$du'_3 = -\frac{C_o e}{2}\left(l_{13}u'_{p,1} + l_{33}u'_{p,3}\right)dt + \left(C_o e dt\right)^{1/2} dx_3 \quad (5)$$

In Eqs. 1-5, the subscript *p* indicates a particle's velocity or position at the previous time step, and dx_i are uncorrelated, normally distributed variables with means of zero and standard deviations of 1. The tensor $l_{ij} = Adj\left(t_{ij}\right)/\det\left(t_{ij}\right) = t_{ij}^{-1}$ is the inverse matrix of the symmetric Reynolds stress tensor t_{ij}. For the idealized case considered here, $t_{13} = u_*^2$, $t_{12} = t_{23} = 0$ and the normal stresses are specified through the following relationships: $s_u = 2u_*$, $s_v = 1.6u_*$ and $s_w = 1.3u_*$. C_o is taken as 5.7 and the dissipation rate is given by $e = u_*^3/kh$, where k is the von Karman constant.

Analytical Gaussian solutions to the conservation equations for a passive scalar continuously emitted from a point source are widely available (e.g., *Seinfeld, J.H. & S.N. Pandis*, 1998). The selection of the plume dispersion parameters have been applied as described by *Singh, B. et al.* (2004). For the test case, 100,000 particles where continuously emitted at a rate *Q* of 100 particles per second from a spherical source of diameter 0.2 m at a height $h = 70$ m. The mean streamwise wind speed was set to $U_1 = 2$ ms^{-1} and $u_* = 0.18$ ms^{-1}. The time step was set to $dt = 1$ s and the duration of the experiment was 1000 s. Concentration averages were calculated over 800 s starting 200 s after the beginning of the release for 8, 100 and 140 collecting boxes used in the *x*, *y* and *z* directions over a domain of 100 m x 100 m x 140 m. The source was located at *x = 20* m, *y = 49*, *z = h =* 70 m.

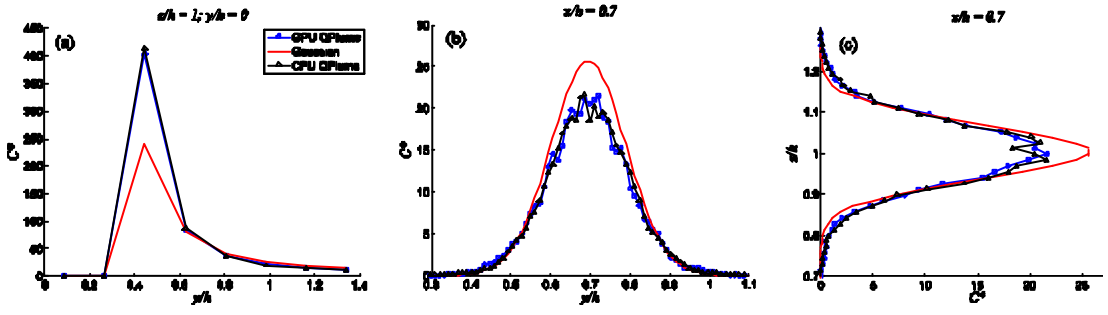

Fig. 1; Concentration comparisons (a) along the plume Centerline, (b) laterally at x/h=0.7and (c) vertically at x/h=0.7.

RESULTS

To compare the Gaussian and QUIC-Plume model runs, the concentrations *C* have been normalized using $C^* = CU_1h^2/Q$. Fig. 1 shows the comparisons between the GPU

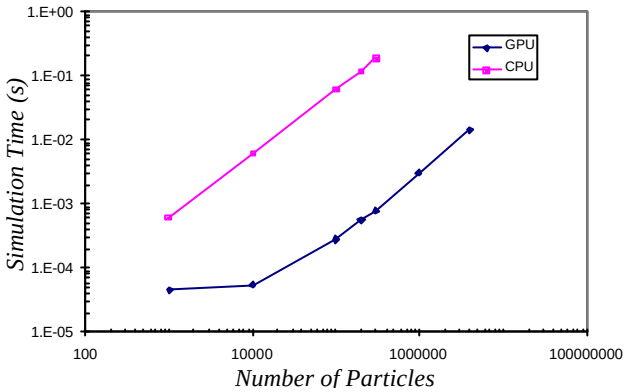

Fig. 2; Average time for GPU & CPU simulations

simulations, the CPU simulations and the Gaussian solution. The GPU and CPU simulations are nearly identical with differences potentially resulting from differences in the random number generators. Single precision computations were used in the CPU and GPU tests since GPUs currently only support single precision. Both simulations over predict concentrations near the source as a result of insufficient collecting box resolution. This issue was addressed by *Singh, B. et al.* (2004) by using a high resolution nested grid near the source.

In addition to the validation exercise, a number of simulations were run over a range of particle numbers to investigate the performance improvement associated with running the simulation on the GPU. Fig. 2 shows the average time required to advect *N* particles for both processors (averaged over 1000 advection steps). A significant performance benefit is gained from the GPU. For simulations with more than 100k particles, a speed up of three orders of magnitude is realized. Note that for the version of QUIC-Plume used in these simulations, the maximum number of particles that could be simulated was just over 300k. Table 1 shows the time required to run a 30 s simulation similar to the test case described above. For this simple test case, approximately 3.3M particles can be simulated in real-time. With the visual display turned on, slightly less than 1 million particles can be simulated in real-time.

Table 1. Time required for a 30 s GPU simulation

N	Time (s)
10,000	0.21
100,172	1.1
1,000,000	8.87
2,000,000	17.0
4,000,000	35.3
6,250,000	55.3

CONCLUSIONS & FUTURE WORK

For this work, we have successfully implemented a GPU version of a random walk model and validated it against a simple dispersion test case. The methodology presented here was applied to the QUIC-Plume model however, it could be used for any type of Lagrangian particle model. The GPU results matched the CPU results quite well and the GPU significantly outperformed the CPU. In addition, it was shown that real-time simulations (and visualization) of large numbers of particles are feasible for simple flows.

We are in the process of optimizing the performance of the GPU-based dispersion model to take better advantage of the GPU architecture. Specifically, making changes that will allow our code to take advantage of Scaleable Link Interface (SLI) GPU configurations. In SLI configurations, multiple graphics cards (in a single PC) distribute the processing load evenly across the GPUs providing a 1.9x speed up when a second card is added. It is currently possible to use up to 4 GPUs with SLI. We also plan to add buildings and higher level Lagrangian models to our code, as well as implement a Red-Black SOR solver to compute the wind field on the GPU. The results from our real-time simulation are being integrated into an immersive virtual environment system for interactive viewing of dispersion simulations.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the financial support of the National Science Foundation (grant IIS–0428856) and Department of Homeland Security.

REFERENCES

Bolz, J., I. Farmer,E. Grinspun & P. Schroder, 2003. Sparse matrix solvers on the GPU: Conjugate gradients and multigrid. ACM Computer Graphics (SIGGRAPH 2003), 917–924.

Galoppo, N., N.K. Govindaraju, M. Henson & D. Manocha, 2005. Lu-gpu: Efficient algorithms for solving dense linear systems on graphics hardware. In Proceedings of the 2005 ACM/IEEE conference on supercomputing, 1–12.

Goodnight, N., Woolley, C., Lewin, G., Luebke, D., & Humphreys, G., 2003. A multigrid solver for boundary value problems using programmable graphics hardware. In Proceedings of the ACM siggraph/eurographics conference on graphics hardware, 102–111.

Hansen, B., Singh, B., Brown, M.J. & Pardyjak, E.R., 2007. Evaluation of the QUIC-URB fast-response urban wind model for an idealized cubic building array, to be submitted to *J. Wind Eng. Ind. Aero.* 2007

Harris, M.J., W.V.I. Baxter, T. Scheuermann, & A. Lastra, 2003. Simulation of cloud dynamics on graphics hardware. In Proceedings of the ACM siggraph/eurographics conference on graphics hardware, 92–101.

Hollerbach, J., D. Grow, & C. Parker, 2005. Developments in locomotion interfaces, 2005 IEEE 9th International Conference on Rehabilitation Robotics, 28 June-1 July 2005, 2005, Chicago, IL, 522-525.

Kipfer, P., M. Sega & R. Westermann, 2004. Uberflow: a gpu-based particle engine. In Proceedings of the ACM siggraph/eurographics conference on graphics hardware, 115-122.

Kirkman, R., M. Deaver, E. Pardyjak & M. Metzger, 2007. Sensitivity Analysis of a Three-Dimensional Wind Tunnel Design, 2006 ASME Joint U.S.-European Fluids Engineering Summer Meeting, FEDSM 2006, July 17-20, Miami, FL, pp. 10.

Kulkarni, S.D., M.A. Minor, M.W. Deaver & E.R. Pardyjak, 2007. Output feedback control of wind display in a virtual environment, Proc. IEEE Intl. Conf. Robotics and Automation, Rome, Italy, April 10-14, 2007.

Moreland, K. & E. Angel, 2003. The FFT on a GPU. In Proceedings of ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics hardware, 112–119.

Nelson, M.A., B. Addepalli, D. Boswell, M.J. Brown, 2006. The QUIC v. 4.5 Start Guide. LA-UR-07-2799.

Pardyjak, E.R. & M.J. Brown, 2001. Evaluation of a fast-response urban wind model–comparison to single-building wind-tunnel data. in Proceedings of the 2001 International Symposium on Environmental Hydraulics. Tempe, AZ.

Pardyjak, E.R. & M. Brown, 2002. Fast response modeling of a two building urban street canyon, 4th AMS Symp. Urban Env., Norfolk, VA.

Rodean, H.C., 1996. Stochastic Lagrangian models of turbulent diffusion, The American Meteorological Society, Boston, MA, pp. 82.

Scheidegger, C., Comba, J., & Cunha, R., 2005. Practical CFD Simulations on the GPU using SMAC. Computer Graphics Forum, 24 (4), 715–728.

Seinfeld, J.H. & S.N. Pandis, 1998. Atmospheric chemistry and physics: from air pollution to climate change, 2nd Ed., Wiley, pp. 1203.

Singh, B., M.D. Williams, E.R. Pardyjak & M.J. Brown, 2004. Development and testing of a dispersion model for flow around buildings. in 4th AMS Symp. Urban Env. Norfolk, VA.

Williams, M.D., M.J. Brown & E.R. Pardyjak, 2002. Development and testing of a dispersion model for flow around buildings. in 4th AMS Symp. Urban Env. Norfolk, VA.