Building Simulation 2009

Eleventh International IBPSA Conference
Glasgow, Scotland
July 27-30, 2009

HIGH-PERFORMACNE AND LOW-COST COMPUTING FOR INDOOR AIRFLOW

Wangda Zuo and Qingyan Chen

National Air Transportation Center of Excellence for Research in the Intermodal Transport Environment (RITE), School of Mechanical Engineering, Purdue University, USA

Corresponding email: yanchen@purdue.edu

ABSTRACT

Computational fluid dynamics (CFD) can provide detailed information of flow motion, temperature distributions and species dispersion in buildings. However, it may take hours or days, even weeks to simulate airflow in a building by using CFD on a single central processing unit (CPU) computer. Parallel computing on a multi-CPU supercomputer or computer cluster can reduce the computing time, but the cost for such high performance computing is prohibitive for many designers. Our paper introduces high performance parallel computing of the airflow simulations on a graphics processing unit (GPU). The computing time can be reduced by 10 - 30 times using the GPU. Furthermore, the cost of purchasing such a GPU is only $500, which is less than 2% of a multi-CPU supercomputer or a computer cluster for the same performance.

INTRODUCTION

To design a comfortable, healthy, and energy-efficient building, it is essential to know some key parameters of the indoor air, such as the distributions of air velocity, air temperature, species concentrations, and pressure. Those data can be obtained by computer simulations (Chen 2009; Davidson 1989). The simulation results should be informative so that the designers can evaluate not only the macro environment in the entire building, but also the microenvironment in each room. The simulations should also be sufficiently fast to explore various alternatives during the design process (Hughes et al. 1994).

By solving the Navier-Stokes equations and other transport equations with an enormous amount of computing nodes, CFD can capture the flow details with good accuracy (Ladeinde and Nearon 1997; Nielsen 2004). However, when the simulated flow domain is large and complex, such as flow in a moderate size building, CFD can be computationally demanding if the simulation is performed on a single CPU computer (Lin et al. 2005; Mazumdar and Chen 2007).

In order to accelerate the CFD simulation, many researchers, such as Crouse et al. (2002) and Mazumdar and Chen (2008), executed simulations in parallel on multi-CPU computers. The parallel computing can greatly reduce the computing time. However, this effort does not reduce the cost for equipment purchase and installation, the space for installing the computers, and the capacity of the cooling system used in the space (Feng and Hsu 2004). Hence, the multi-CPU computing is luxury for building designers. It is necessary to find high-performance and low-cost computing hardware for simulating flow in buildings.

Recently, GPU has attracted attention for parallel computing. Different from CPU, GPU is the core of a computer graphics card, which integrates multiple streaming processors on a chip. The GPU structure is highly parallelized for high performance graphics processing. For example, a NVIDA GeForce 8800 GTX GPU available in 2006 integrated 128 processors so that its peak computing speed is 367 GFLOPS. Comparatively, the peak performance of an INTEL Core2 Duo 3.0 GHz CPU available at the same time is only about 32 GFLOPS (Kirk and Hwu 2008). Figure 1 compares peak performance of the CPU (INTEL) and the GPU (NVIDIA). The performance gap between the CPU and GPU has been expanding since 2003 (NVIDIA 2007). Furthermore, this trend is likely to continue in the future. Besides its high performance, the cost of a GPU is low. For example, a graphics card with NVIDIA GeForce 8800 GTX GPU costs only around $500 and it can easily be installed into a personal computer.

CPU handles sequential jobs so that it increases the computing speed principally by increasing its clock frequency. Unlike CPU, GPU is normally used for graphics processing that is typically a parallel job. Thus, development of GPU is to increase its computing capacity by adding more processors to handle the parallel job. It is technically easy and economically inexpensive to integrate a large quantity of low frequency processors into one chip (Kirk and Hwu 2008). Having many low-clock-frequency processors working in parallel, GPU can achieve a high computing speed. For example,

although the clock frequency of the NVIDIA GeForce 8800 GTX GPU is only 575 MHz, its computing speed can be as high as 367 GFLOPS with 128 processors. The development strategy of GPU makes a graphics card at low-cost and with high-performance.

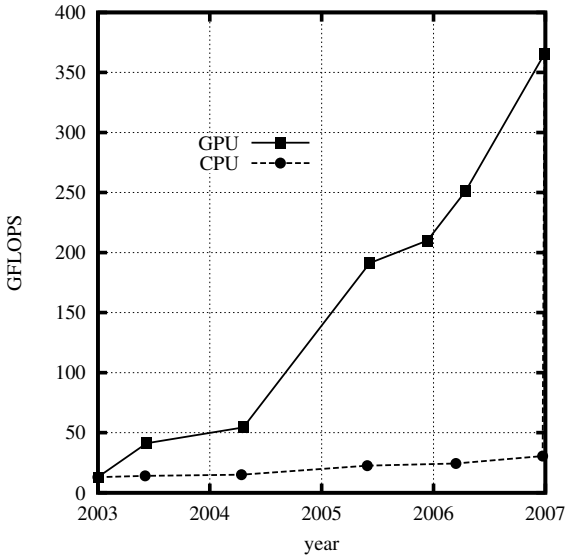


Figure 1 Comparison of computing speed of GPU and CPU

In spite of its advantages in cost and performance, the applications of GPU have been focused on image processing. The programming on GPU requires a deep understanding of its hardware and programming languages. Prior to 2006, GPU programming languages, such as OpenGL (Shreiner and OpenGL Architecture Review Board 2008) and Direct3D (Walsh 2006), were designed for graphics processing. It was difficult to use the languages for solving Navier-Stokes equations. Hence, only a few attempts were made by experts in graphics (Ho et al. 2008; Scheidegger et al. 2005; Wei et al. 2004).

In 2006, NVIDIA (NVIDIA 2007) provided a new GPU programming environment, named computer unified data architecture (CUDA). CUDA is an extended C language. Like other advanced programming languages, CUDA allows users to manipulate GPU without knowing the details of the hardware. Furthermore, CUDA is compatible with the standard C language. If a flow simulation code is written in C, a user only needs to rewrite the parallel computing part in CUDA. This feature can save a lot of time on code development. Some researchers have started to use CUDA for GPU programming. For example, Rodrigues et al (2008) used it for molecular simulations and Manaveki (2007) for cryptography. They have made the simulations 10-20 times faster than those on a CPU. Thus, it is also interesting to perform indoor flow simulations using the FFD on GPU.

IMPLEMENTATION

Flow Model

This investigation applied a fast fluid dynamics (FFD) model proposed by Stam (1999). It is a simplified CFD model for solving continuity equation, Navier-Stokes equations and transport equation for energy and species concentrations for transient, incompressible fluid flow. To efficiently solve these partial differential equations, FFD splits them into many simple equations and solves the simple equations one by one. The current FFD model is first order in time and second order in space. Applying the FFD model, Zuo and Chen (2009) simulated different indoor airflows at a speed 50 times faster than a CFD model. This investigation used the FFD model for flow simulation on GPU.

Software and Hardware

The implementation used CUDA to divide a GPU into three levels (Figure 2). The highest level is "grid". Each grid consists of multiple "blocks", and every block has many "threads". A thread is the basic computing unit of GPU. Mathematic and logic operations are performed on threads.

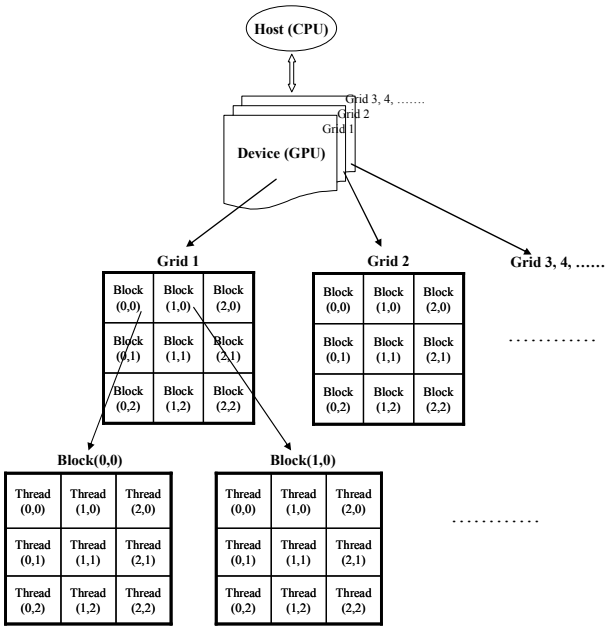


Figure 2 Schematic of parallel computing with CUDA

This study used a NVIDIA GeForce GTX 8800 GPU. This GPU has 16 streaming multiprocessors (SMs) (Rixner 2002) and each SM can hold up to 8 blocks or 768 threads at one time. Thus, the entire GPU can simultaneously hold up to 12,288 threads. Because CUDA does not allow splitting a block into two SMs, the block assignment is crucial to realize the full capacity of GPU. For example, if a block has 512 threads, then only one block can be assigned to one

SM and the remaining 256 threads in that SM are unused. If a block contains 256 threads, then three blocks can fully occupy all of the 768 threads of an SM. Theoretically, the 8800 GTX GPU can reach its peak performance when all 12,288 threads are running at the same time. Practically, the peak performance also depends on many other factors, such as the time for reading or writing data with the memory.

Mapping Strategy

When working in parallel, it is important to map the thread indices (*threadID.x*, *threadID.y*) in a block onto the coordinate of mesh nodes (*i, j*). The current implementation applied the following formulas:

$$i = blockDim.x \times blockID.x + threadID.x \,, \quad (1)$$

$$j = blockDim.y \times blockID.y + threadID.y \,. \quad (2)$$

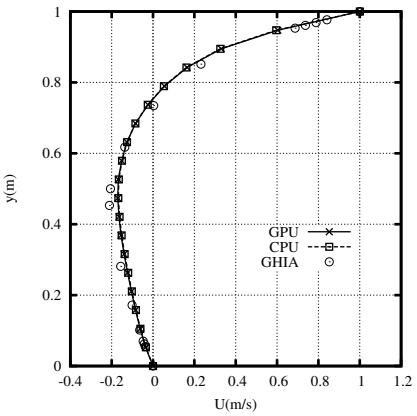
where *blockID.x* and *blockID.y* are the indices of the block for the thread. *blockDim.x* and *blockDim.y* are the block dimensions at *x* and *y* directions, respectively. Both of them are 16 in our implementation.

SIMULATION RESULTS

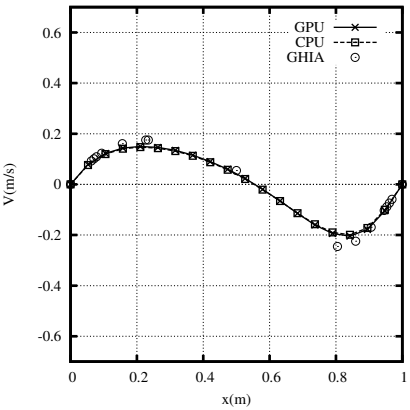
To demonstrate FFD simulations on GPU, this investigation calculated three airflows relevant to indoor environment and compared the results with those on CPU and the data from literature. The three flows were laminar and turbulent flow in a lid-driven cavity and natural convective flow in a tall cavity. The simulations used the exactly same meshes and numerical settings in both GPU and CPU versions.

Laminar Flow in a Lid Driven Cavity (*Re* = 100)

The first case was the laminar flow in a lid-driven cavity. Based on the length of cavity and lid velocity, the Reynolds number of the flow is *Re* = 100. This study used a mesh of 33 × 33 grids. The reference data was the high quality CFD results obtained by Ghia et al (1982). As shown in Figure 3, FFD on GPU could predict the same velocity profiles as that on CPU. Furthermore, the FFD results were similar to the reference data. Although this is a simple case, it proves that GPU can be used for numerical calculations.



(a) Horizontal velocity at $x = 0.5\ L$

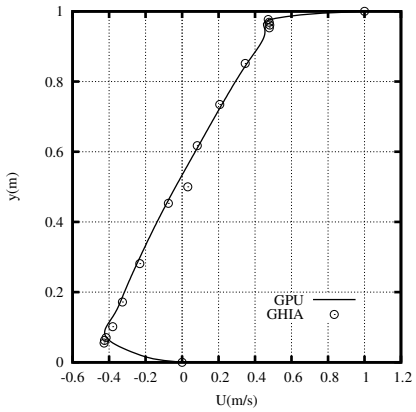


(b) Vertical velocity at y = 0.5 *L*

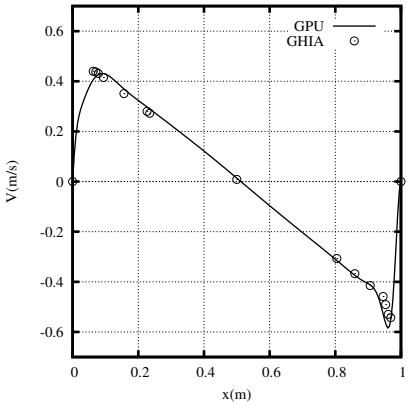
Figure 3 Comparison of the calculated velocity profiles (*Re* = 100) by the FFD model on CPU and GPU with the CFD data in a lid-driven square cavity

Turbulent Flow in a Lid Driven Cavity (*Re* = 10000)

The second case was a turbulent flow in the lid-driven cavity with *Re* = 10000. To capture the characteristics of the flow, this study used a very fine mesh of 513 × 513 grids. The same amount of mesh was also used by Ghia et al. (1982). Because FFD is the first order in time, a very small time step size (0.005s) was necessary to reduce the error. It is very time consuming to run such an unsteady simulation on a single CPU. Thus, this study only did the simulation on GPU. As shown in the Figure 4, The FFD model on GPU computed accurately the horizontal and vertical velocity profiles. The computed profiles agree with the reference data obtained by Ghia et al. (1982).



(a) Horizontal velocity at $x = 0.5\ L$



(b) Vertical velocity at y = 0.5 L

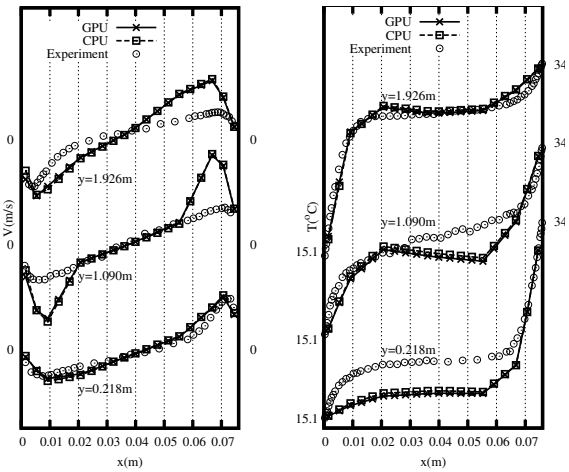
Figure 4 Comparison of the calculated velocity profiles (Re = 10000) by the FFD model on GPU with the reference data in a lid-driven cavity

Figure 5 shows streamlines computed by the FFD model on GPU for the turbulent flow. The FFD model on GPU can properly predict a large recirculation in the center of cavity. It also computed several secondary recirculations at low-left, low-right and upper-left corners. Furthermore, it captured two third recirculations at the low-left and low-right corners. The differences between the FFD prediction and reference data (Ghia et al. 1982) are very negligible.

Natural Convection in a Tall Cavity

The flows in the previous two cases were isothermal. The FFD model on GPU was further tested for a non-isothermal flow. The non-isothermal flow was a natural convection flow in a tall cavity of 0.076 m wide and 2.18 m high. The left wall was cooled at 15.1 ^{o}C and the right wall heated at 34.7 ^{o}C. The top and bottom walls were insulated. The corresponding Rayleigh number was 0.86×10^{6}. A coarse mesh of 11 × 21 was applied to the FFD simulations on both CPU and GPU. The reference data was from the experiment performed by Betts and Bokhari (2000). Figure 6 depicts that the FFD model on GPU gave the same velocity and temperature profiles as that on CPU. Although the results obtained by the FFD model differ from the experimental data, the error was caused by the FFD model, not GPU.

Figure 5 Streamlines of a turbulent lid-driven cavity flow (Re = 10000) computed by the FFD on GPU



(a) Velocity profiles (b) Temperature Profiles

Figure 6 Comparison of the velocity and temperature profiles predicted by the FFD model on CPU and GPU with the experimental data

DISCUSSION

Computing Time

To compare the FFD simulation speed on GPU with that on CPU, this study used the computing time for the lid-driven cavity flow at Re = 100 as an example. The CPU simulations were conducted on a HP workstation with an INTEL XeonTM CPU and the GPU simulations on an NVIDIA GTX 8800 GPU. The simulations were performed for 100 time steps but with different meshes.

Figure 7 illustrates that the CPU computing time increased linearly with the mesh size. When the grid

number was smaller than 3.6×10^3, the FFD model on CPU was faster than that on GPU. Since it took time to transfer data during the GPU simulations, the time could be more significant than that saved in the parallel computing when the mesh size was small. Hence, parallel computing on GPU should be applied to cases with a large mesh size.

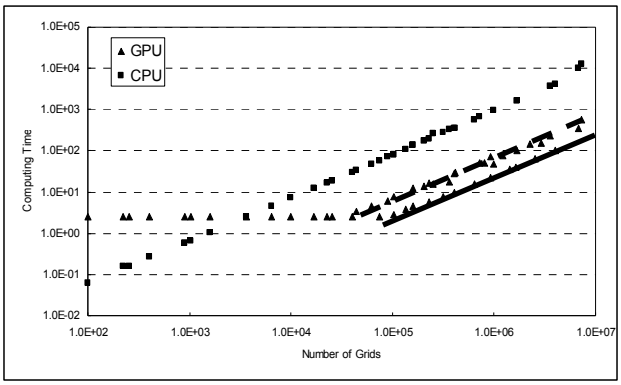


Figure 7 Comparison of the computing time used by the FFD model on GPU with that on CPU

It should be noted from Figure 7 that the GPU computing time was almost constant when the mesh size was less than 4×10^4. This is because the mesh size was not large enough for the GPU utilizing fully its capacity. When the mesh size was greater than 4×10^4, the GPU computing time increased along two paths. Those points on the solid line were for the cases with a mesh size in multiplication of 256 and on the dashed line the mesh size not in multiplication of 256. As mentioned previously, each mesh node was assigned to one thread and a block had 256 threads. If the mesh size was in the multiplication of 256, all the 256 threads of every block were utilized. Thus, the working load among the blocks was equal. Otherwise, some of the threads in the block were rendered idle and the working load between the blocks was unequal. The imbalance of the working load can have a severe penalty on the computing speed. For example, the simulation with 640 × 640 grids that was in multiplication of 256 took 9.977 s, but that with 639 × 639 needed 28.875 s. Although the latter case had fewer grids than the former, its computing time increased by almost two times.

Nevertheless, the FFD model on GPU is still 10 times faster than that on CPU even if the grid number was not in multiplication of 256. The difference increased to around 30 times if the grid number was in multiplication of 256.

Impact

This study implemented the FFD model for flow simulations on GPU. Since the FFD model solves the same governing equations as the CFD model does, it is also possible to implement the CFD model on GPU. One can also expect that the speed of CFD simulations on GPU should be faster than that on the CPU. For those CFD codes written in C language, the implementation will be relatively easy since only the parallel computing part needs to be re-written in CUDA.

It is possible to further reduce the computing time by using multi-GPU clusters. For example, the NVIDA Tesla personal supercomputer has 4 GPUs with 960 processors. It is about 250 times faster than a single CPU personal computer.

CONCLUSION

This study performed flow simulation with the FFD model on GPU and CPU. The FFD simulation on GPU is 10 – 30 times faster than that on CPU. The cost of a GPU is less than 2% of a supercomputer or computer cluster with the same performance. The GPU can be used also for CFD simulations and other scientific computing.

ACKNOWLEDGEMENT

This project was funded by U.S. Federal Aviation Administration (FAA) Office of Aerospace Medicine through the National Air Transportation Center of Excellence for Research in the Intermodal Transport Environment (RITE) Cooperative Agreement 04-C-ACE-PU-002. Although the FAA has sponsored this project, it neither endorses nor rejects the findings of this research. The presentation of this information is in the interest of invoking technical community comment on the results and conclusions of the research.

REFERENCES

Betts, P.L. and Bokhari, I.H. (2000) "Experiments on turbulent natural convection in an enclosed tall cavity", *International Journal of Heat and Fluid Flow,* **21**, 675-683.

Chen, Q. (2009) "Ventilation performance prediction for buildings: A method overview and recent applications", *Building and Environment,* **44**, 848-858.

Crouse, B., Krafczyk, M., Kuhner, S., Rank, E., and Van Treeck, C. (2002) "Indoor air flow analysis based on lattice Boltzmann methods", *Energy and Buildings,* **34**, 941-949.

Davidson, L. (1989) *Numerical simulation of turbulent flow in ventilated rooms*, Ph. D. Thesis, Chalmers University of Technology, Goeteborg, Sweden.

Feng, W. and Hsu, C. (2004) "The origin and evolution of green destiny", In: *Proceedings of IEEE Cool Chips VII: An International*

Symposium on Low-Power and High-Speed Chips, Yokohama, Japan.

Ghia, U., Ghia, K.N., and Shin, C.T. (1982) "High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method", *Journal of Computational Physics,* **48**, 387-411.

Ho, T.Y., Lam, P.M., and Leung, C.S. (2008) "Parallelization of cellular neural networks on GPU", *Pattern Recognition,* **41**, 2684-2692.

Hughes, J., King, V., Rodden, T., and Andersen, H. (1994) "Moving out from the control room: Ethnography in system design", In: *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, Chapel Hill, North Carolina, pp. 429-439.

Kirk, D. and Hwu, W.-M.W. 2008. Lecture notes of programming massively parallel processors: University of Illinios, Urbana-Champaign.

Ladeinde, F. and Nearon, M.D. (1997) "CFD applications in the HVAC&r industry", *ASHRAE Journal,* **39**, 44-48.

Lin, C., Horstman, R., Ahlers, M., Sedgwick, L., Dunn, K., and Wirogo, S. (2005) "Numerical simulation of airflow and airborne pathogen transport in aircraft cabins - part 1: Numerical simulation of the flow field", *ASHRAE Transactions,* **111**.

Manavski, S.A. (2007) "CUDA compatible GPU as an efficient hardware accelerator for aes cryptography", In: *Proceedings of 2007 IEEE International Conference on Signal Processing and Communications (ICSPC 2007)*, Dubai, United Arab Emirates.

Mazumdar, S. and Chen, Q. (2007) "Impact of moving bodies on airflow and contaminant transport inside aircraft cabins", In: *Proceedings of ROOMVENT 2007*, Helsinki, Finland, pp. 13-15.

Mazumdar, S. and Chen, Q. (2008) "Influence of cabin conditions on placement and response of contaminant detection sensors in a commercial aircraft", *Journal of Environmental Monitoring,* **10**, 71-81.

Nielsen, P.V. (2004) "Computational fluid dynamics and room air movement", *Indoor Air,* **14**, 134-143.

NVIDIA. (2007) *NVIDIA CUDA compute unified device architecture-- programming guide (version 1.1)*, Santa Clara, California, NVIDIA Corporation.

Rixner, S. (2002) *Stream processor architecture*, Boston & London, Kluwer Academic Publishers.

Rodrigues, C.I., Hardy, D.J., Stone, J.E., Schulten, K., and Hwu, W.-M.W. (2008) "GPU acceleration of cutoff pair potentials for molecular modeling applications", In: *Proceedings of the 2008 International Conference on Computing Frontiers*, New York, pp. 273-282.

Scheidegger, C.E., Comba, J.L.D., Da Cunha, R.D., and Corporation, N. (2005) "Practical CFD simulations on programmable graphics hardware using SMAC", *Computer Graphics Forum,* **24**, 715-728.

Shreiner, D. and OpenGL Architecture Review Board. (2008) *OpenGL programming guide: The official guide to learning OpenGL (version 2.1)* (6th ed.), Upper Saddle River, New Jersey, Addison-Wesley.

Stam, J. (1999) "Stable fluids", In: *Proceedings of 26th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'99)*, Los Angeles, pp. 121-128.

Walsh, P. (2006) *Advanced visual effects with Direct3D*, Boston, Massachusetts, Course Technology.

Wei, X., Li, W., Mueller, K., and Kaufman, A.E. (2004) "The Lattice-Boltzmann method for simulating gaseous phenomena", *Ieee Transactions on Visualization and Computer Graphics,* **10**, 164-176.

Zuo, W. and Chen, Q. (2009) "Real-time or faster-than-real-time simulation of airflow in buildings", *Indoor Air,* **19**, 33-44.