A first approach to simple and faster 2D flood model in GPGPU enviroment

Adiel S. Sarates Jr
Federal University of Rio Grande do Sul
Parallel and Distributed Processing Group
Porto Alegre, Rio Grande do Sul, Brazil
adiel.seffrin@inf.ufrgs.br

Abstract

A third of global population lives near of the coast, large urban centers have problems with their flow system and all of this creates concern in flood events. Therefore, simple and faster models are increasingly required. The current models tend to have instability points and new equations are being created from the quasi-linearized one-dimensional SaintVenant or Shallow Water equations. Normally, flood models are widely parallelizable, being GPGPU an interesting enviroment to explore model performance. Related works shows that the expected speedups are satisfactory. This paper shows initial results of a CUDA version of this new equations, analising the speedup and the estimated error compared to the traditional FORTRAN version.

1. Introduction

With more than 29% of the wolrd's population living within 60Km of the coast [1], flood events are a constant concern of governments and nations. The effects of inundations vary from wetlands until entire cities with underwater houses. These events generate economic and social impacts, a high costin aid programs in addition to the infrastructure damages. Warning systems and action plans in case of flooding are already part of the daily life of dozens countries.

Based on this flood inundation models have become an integral part of flood risk assessment, providing means to predict flood impact analysing the expected water depth or the flow velocity, etc. These models vary in complexity from solutions of the two-dimensional shallow water equations [6] to storage cell models based on Manning's equation [3]. Commonly these models are expensive for many real world applications and to have a more effective approach, they have begun to develop the program in a way it will be able to run on parallel. Some models are widely parallelizable and to take advantage of this, graphics processor units (GPU) have been used [2].

Since first proposed methods [8] to predict flood plain inundation using storage cell approaches have become popular.

Initially, such methods discretized floodplains into irregular polygonal units representing large (surface areas of $10^0 10^1$ km^2) natural storage compartments and calculated the fluxes of water between these according to some uniform flow formula such as the weir or Mannings equations. For many such models in-channel flows are calculated using some form of the 1D SaintVenant equations, and when bankfull flow is exceeded water is routed into and between the floodplain storage units. More recently the availability of increased computing power and detailed descriptions of floodplain topography available has allowed a move away from large, irregular storage units to the discretization of the floodplain as a fine spatial resolution regular grid (cell areas of $10^{-2} 10^{-3}$ km^2). Here each cell within the grid is a storage area for which the mass balance is updated at each time step according to the fluxes of water into and out of each cell. Similar to polygonal storage cell models, fluxes are calculated analytically using uniform flow formulae but with the advantage of higher resolution predictions and removal of the need for the modeller to make explicit decisions about the location of storage compartments and the linkages between these. [7]

1.1. A serial approach

Numerous such models solve a continuity equation relating flow into a cell and its change in volume and a equation for each direction where flow between cell is calculated according to Manning's law:

$$Q_x^{i,j} = \frac{h_{flow}^{5/3}}{n} \left(\frac{h^{i-1,j} - h^{i,j}}{\Delta x} \right)^{1/2} \Delta y \qquad (1)$$

where $h^{i,j}$ is the water free surface height [L] at the node (i, j), Δ_x and $Delta_y$ are the cell dimensions [L], t is the time [T], n is the Mannings friction coefficient $[L^{-1}/3T]$, h_{flow} represents the depth through which water can flow between two cells, and is defined as the difference between the highest water free surface in the two cells and the highest bed elevation and Q_x and Q_y describe the volumetric flow rates between floodplain cells [L3 T^{-1}]. Q_y is defined analogously to Eq. (1).

The advantage of the storage cell formulation is that fluxes are calculated analytically so the computational costs per time step are potentially much lower than in equivalent numerical solutions of the full shallow water equations. Althought, unless the constant time-step used to generate h_{flow} was small, simulations quickly developed a type of instability as all the water in a particular cell drained into adjacent ones in a single (large) time step. At the next time step, this situation would reverse and all the water would flow back. To solve this problem was introduced some kind of flow limiter who sets the maximum flow that can occur between cells and is typically a function of flow depth, grid cell size and time step. To not use the 'flow limiter', Hunter et al. (2005) [4] provided a solution based on adaptive time-stepping. This approach seeks to remove the need to invoke the flow limiter by finding the optimum time step at each iteration. The adaptive time step model showed a better absolute performance than the classical fixed time-step version at this spatial resolution, but at approximately six times the computational cost. In particular the adaptive model appeared able to simulate floodplain wetting and drying more realistically. Despite this success, the results obtained by Hunter et al. (2006) [**?**] identified a fundamental problem with the equation used, namely that the optimum stable time step reduces quadratically with decreasing grid size, so adaptive time step storage cell codes are therefore incompatible with the fine spatial resolution grids increasingly required for urban flood modelling.

It was clear that to allow wide area urban flood modelling at fine spatial resolution a new hydraulic model formulation was required. Based on this Bates et al. (2010) [7] development and testing a set of flow equations for adaptive time step storage cell models which can overcome the quadratic dependency on grid size yet which can be solved analytically with approximately the same computational cost as Eq. (1). The new scheme therefore retains all the computational advantages of storage cell models over full 2D codes whose equations require expensive numerical solution, yet with none of the previous disadvantages.

1.2. A new formulation of the shallow water equations

The starting point for derivation of such an equation is therefore the momentum equation from the quasi-linearized one-dimensional SaintVenant or Shallow Water equations and with some transformations [7] was obtained a explicit form for calculation of flows at the new time-step in the model:

$$q_{t-\Delta t} = \frac{q_t - gh_t\Delta t \frac{\delta(h_t+z)}{\delta x}}{\left(1 + gh_t\Delta t n^2 q_t / h_t^{10/3}\right)} \qquad (2)$$

The enhanced stability of Eq. (2) stems from the increase in the denominator as the friction term increases, forcing the flow to zero, as would be expected for shallow depths. Unlike (1), Eq. (2) includes shallow water wave propagation so while stability is improved, it is still subject to the CourantFreid-richsLevy condition where the non-dimensional Courant number, needs to be less than 1 for stability. This gives a necessary but not sufficient condition for model stability and is used to estimate a suitable model time-step at $t + \Delta t$:

$$\Delta t_{max} = \alpha \frac{\Delta x}{\sqrt{gh_t}} \qquad (3)$$

where α is a coefficient used to produce a stable simulation and is included because the Courant number is not sufficient to ensure model stability.

2. Problem

The first version of Bates' model [7] consists in some matrix of data for water flow, the speed of this flow, the depth of water and the height of water including the terrain's elevation under it. The model was developed in a serial FORTRAN code which calculate for each time-step in each iteration, the discharge of water between cells and apply the continuity equation on to keep the heigh of water continuous. The aim of this work is generate a parallel version of this model using NVIDIA CUDA and analyze his behavior in this enviroment and also find an optimal solution considering a high speedup and the lower error caused by precision difference.

2.1. Implementation

Both discharge and continuity function must be applied all over the cells of grid, using the data from previously time-step to update them, thus both function could be write as Cuda kernels. In this way, based on dimention of the block and grid and the index of block and thread, each thread 'know' what row and column must to be uptaded. In each kernel are calculated the new values for the cells in interior of grid and after this, the new values from the border, respecting some boundary conditions. The iteraction control was made by CPU, as well as the *while* clause - who invokates kernels -, getting the updated value of time-step from GPU when all threads was finished the last kernel call.

When an iteraction end, the values of heights and depths of water, flow of discharges and velocities in x nad y directions was be writed on HD.

2.2. First Results

For a small instance of the problem (100 rows and 100 columns), was obtained a speedup of 17x using a GTX480 to execute the kernels instead of a Core2 Duo E8500 @ 3.16GHz where the maximum error of a water flow (Q-Error) was 0.35. Table 1 shows the obtained times.

Language	Cells	Time	SpeedUp	Q-Error
Fortran	8.1k	227 secs	1x	-
CUDA	8.1k	15 secs	$\approx$ 15x	0.02
Fortran	10k	387.79 secs	1x	-
CUDA	10k	22 secs	$\approx$ 17x	0.35
Fortran	12.1k	632 secs	1x	-
CUDA	12.1k	29 secs	$\approx 21x$	>300

Table 1. Comparison of times of Fortran vs Cuda.

2.3. Issues to solve

When the gridsize increases, the parallelized version shows a instability for higher iteraction generating a model divergence and some cell values has infinite flow of water. Further, the modern models works with grids in range of 3k to 3M [5] cells and at the present moment only small intances ($\approx 10k$ cells) was acceptably treated in this work.

3. Conclusion and future work

The quickly response for flood events is a leading factor to prevent disaster, thus parallelized models have an important role to make this feasible. The new approach using a derivation from quasi-linearized one-dimensional SaintVenant or Shallow Water equations has produced good results. In initial experiments, a good speedup was obtained with a acceptable error making this method promising. Other models have been optimized with help of GPGPU and found significant improvments.

The parallelized version of these new equations, shows an instability case when the grid size increases. The difference of precision between CUDA and FORTRAN may cause this problem and must be researched, as an unexpected concurrent memory access. Correction algorithms are being studied to minimize the precision errors effects and some modifications in the model, also to improve the GPGPU's effects are not discarded.

References

[1] Gommes, R., du Guerny, J., Nachtergaele, F., Brinkman, R., 1997. Potential Impacts of Sea-level Rise on Populations and Agriculture. Food and Agricultural Organization of the United Nations, Rome. Available at: ¡http://www.fao.org/sd/eidirect/EIre0045.htm¿.

[2] Kalyanapu, A. J., Shankar, S., Pardyjak, E. R., Judi, D. R., & Burian, S. J. (2011). Assessment of GPU computational enhancement to a 2D flood model. Environmental Modelling & Software, 26(8), 1009-1016.

[3] Bates, P.D., De Roo, A.P.J., 2000. A simple raster-based model for flood inundation simulation. Journal of Hidrology 236(1-2), 54-77.

[4] Hunter, N.M., Horritt, M.S., Bates, P.D., Wilson, M.D., Werner, M.G.F., 2005. An adaptive time step solution for raster-based storage cell modelling of floodplain inundation. Advances in Water Resources 28, 975991

[5] Neal, J. C., Fewtrell, T. J., Bates, P. D., & Wright, N. G. (2010). A comparison of three parallelisation methods for 2D flood inundation models. Environmental Modelling & Software, 25(4), 398-411.

[6] Villanueva, I., Wright, N.G., 2006 Linking Riemann and storage cell models for flood Prediction. ICE Journal of Water Management 159, 27-31

[7] Bates, P.D., Horritt, M.S., Fewtrell, T.J, 2010. A simple inertial formulation of the shallow water equations for efficient two-dimensional flood inundation modelling. Journal of Hidrology 387, 33-45

[8] Zanobetti, D., Longer, H., Preissmann, A., Cunge, J.A., 1970. Mekong deltamathematical model program construction. American Society of Civil Engineers, Journal of the Waterways and Harbors Division 96 (WW2), 181199.