Computers & Geosciences 62 (2014) 178–186

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Accelerating a hydrological uncertainty ensemble model using graphics processing units (GPUs)

D. Tristram [a,*], D. Hughes [b], K. Bradshaw [a]

[a] Department of Computer Science, Rhodes University, Grahamstown 6139, South Africa
[b] Institute for Water Research, Rhodes University, Grahamstown 6139, South Africa

ARTICLE INFO

Article history:
Received 27 January 2013
Received in revised form
5 July 2013
Accepted 15 July 2013
Available online 12 August 2013

Keywords:
GPGPU
OpenCL
Hydrological modelling

ABSTRACT

The practical application of hydrological uncertainty models that are designed to generate multiple ensembles can be severely restricted by the available computer processing power and thus, the time taken to generate the results. CPU clusters can help in this regard, but are often costly to use continuously and maintain, causing scientists to look elsewhere for speed improvements. The use of powerful graphics processing units (GPUs) for application acceleration has become a recent trend, owing to their low cost per FLOP, and their highly parallel and throughput-oriented architecture, which makes them ideal for many scientific applications. However, programming these devices efficiently is non-trivial, seemingly making their use impractical for many researchers. In this study, we investigate whether redesigning the CPU code of an adapted Pitman rainfall-runoff uncertainty model is necessary to obtain a satisfactory speedup on GPU devices. A twelvefold speedup over a multithreaded CPU implementation was achieved by using a modern GPU with minimal changes to the model code. This success leads us to believe that redesigning code for the GPU is not always necessary to obtain a worthwhile speedup.

1. Introduction

The history of the development and practical application of hydrology, water resources, and other environmental models has been closely aligned with the development of computer hardware and software. While there has always been a focus on the scientific aspects of model improvements (O'Connell, 1991; Beven, 2001), there have also been constraints on what can be achieved given the available computing power. In the early days of model development (1960–1970s), access to mainframe digital computers was largely restricted to universities or large state agencies, mainly owing to the associated cost of the hardware. Even those computers were relatively slow and lacked the memory that we have become accustomed to in modern computers. Consequently the developments made within research institutions were not generally available to practising scientists and engineers. Well into the 1980s and 1990s, many proposed scientific developments remained constrained by computing limitations and were still not generally available for practical use. These developments included the use of finer time intervals in models (Hughes and Sami, 1994), increased spatial resolution through fully distributed models (Beven, 1993; Refsgaard, 1997), automatic parameter calibration approaches (Madsen, 2000), as well as integrating models of different types (Argent and Houghton, 2001).

With the increased availability and use of desktop computers (PCs) through the 1980s, it was possible for environmental models to become part of the everyday toolbox of scientific and engineering practitioners. However, it was only around 2005 that it became practical to run complex, spatially detailed models on PCs. Arguably, there always seems to be a need for greater computing power in environmental modelling despite the increases in power that have become available on a regular basis. As one type of model development becomes achievable through hardware and software improvements, others are conceived that stretch the limits of the available technology. The most recent advance in hydrological modelling is the use of uncertainty approaches (Beven and Binley, 1992; Freer et al., 1996; Vrugt et al., 2003; Pappenberger and Beven, 2006; Hughes et al., 2010), which involves (in many different ways) running a model many 1000's of times and generating ensembles of outputs rather than a single solution. As with most automatic calibration approaches (Yapo et al., 1998), the amount of computing time is considerable, such that even using modern PCs (with conventional software architecture), uncertainty approaches to modelling largely remain part of academic research and have not found a place in practice.

Computer processors are continuously and reliably getting faster and more complex. As well as the drive for scientific advancement, the demand for faster processors can be attributed to the ever increasing computation requirements of the research,

* Corresponding author. Tel.: +27 46 603 8291; fax: +27 46 636 1915.
E-mail addresses: d.tristram@boost.za.net (D. Tristram), d.hughes@ru.ac.za (D. Hughes), k.bradshaw@ru.ac.za (K. Bradshaw).

0098-3004/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cageo.2013.07.011

industrial, business, and entertainment sectors. In the past, computer users simply waited for reliable increases in processor speeds to handle computation problems that were not really feasible at the time. However, it is believed that single-core processors have hit the *power wall* (Meenderinck and Juurlink, 2009), meaning that single-core frequency improvements can no longer easily be made because of power and heat constraints (Ross, 2008). CPU manufacturers have thus changed their focus from single-core processors to multicore processors. The last generation Pentium 4, released in 2006, was Intel's final flagship single-core desktop CPU. What followed were CPUs with more features and more cores, starting with Intel's Core 2 Duo featuring two cores. Modern desktop CPUs have up to eight cores. The trend of increasing CPU core counts means that existing software needs to be redesigned to take advantage of multiple CPU cores. In the case of uncertainty modelling, additional cores are beneficial since the computations are parallel in nature, but there is still an insufficient number of cores on a modern CPU to perform an uncertainty analysis of a moderately complex model within a "reasonable" amount of time.

In contrast to modern CPUs, which follow the multiple instruction, multiple data architecture, graphics processing units (GPUs) are built as massively parallel processors using the single instruction, multiple data (SIMD) architecture. Driven primarily by the computer gaming industry, commodity GPUs have become powerful and highly parallel computation devices. In recent years, their increased programmability has made them attractive for general purpose computation, and GPU manufacturers have been improving the ability of GPUs to perform such computations efficiently. Arguably, a solution to existing efficiency problems in uncertainty environmental modelling is the use of powerful GPUs as cost-effective accelerators for problems that map well to a SIMD architecture. However, efficient GPU programming is non-trivial (Ryoo et al., 2008; Daga et al., 2011; Sim et al., 2012), which seemingly restricts the benefits of general-purpose computation on graphics processing units (GPGPU) to individuals experienced in the field of GPU programming.

This paper documents a study which explores whether redesigning a CPU-based scientific model for efficient execution on GPUs is strictly necessary to obtain a worthwhile speedup. We consider a worthwhile speedup as one that increases the convenience of running the model with the desired parameters, increases work efficiency, and allows the modeller to start thinking about other improvements that can be made to extend the benefits of the model. The hydrological simulation model chosen is an uncertainty version (Hughes et al., 2010) of the Pitman (Pitman, 1973; Hughes, 2004) rainfall-runoff hydrological model that is widely used in the southern Africa region for both research and practical water resources assessments. However, the results of the study should be applicable to any environmental model with a similar software structure.

2. Pitman hydrological model

The Pitman model is a conceptual type, monthly time-step, semi-distributed (sub-catchment) model that includes some 23 parameters that govern the algorithms defining the hydrological storages and processes such as evapotranspiration, interception, surface runoff, soil moisture storage, interflow, groundwater recharge and drainage, and catchment routing. An overview of the hydrological processes and their relationships for this version of the model is illustrated in Fig. 1. The full details of the model are not given here as the study could have used any model of this type. The conceptual diagram in Fig. 1 and the brief explanation of the model are merely provided to illustrate the degree of model complexity. The model is typically run over a period of 40–90 years (480–1080 months) depending on the availability of input rainfall data. Each component of the model (Fig. 1) consists of a set of sequential algorithms that generate either output data or the values of internal state variables that are used in the next time interval or as input to the next downstream sub-catchment. Most of the model components operate over four equal steps within the one-month main time step to avoid excessive changes in any of the state variables (storages or fluxes) before other components are updated. This approach is frequently used in coarse time step models (Hughes, 2004) in recognition of the fact that, in nature, water balance components operate simultaneously.

The ability of the model to accurately represent the hydrological response of any given catchment is reliant on the correct specification of the model parameters. Estimation of these parameters is always problematic, even if they are calibrated against an observed stream flow time series. Many of the parameter estimation issues are associated with inter-relationships between model parameters and the problem of equifinality (Beven, 2006), whereby similar model outputs can be achieved with different parameter sets.

The uncertainty version of this model is designed to assist in the estimation of these parameters and allows the model results for many different options within the feasible parameter space to be explored (Hughes et al., 2010). It has the goal of establishing parameter values, setting parameter uncertainty bounds (Kapangaziwiri et al., 2012), and exploring parameter inter-dependencies. Parameter inputs to the model are specified as either means and standard deviations of normal distribution functions, or minimum and maximum values of uniform distribution functions.

The Delphi code on which this model is based has evolved from the first version of the Spatial and Time Series Information Modelling (SPATSIM) system developed in the early 2000s (Hughes and Forsyth, 2006), rather than having been meticulously designed from the perspective of efficient software architecture. The model is run many times (typically between 5000 and 20,000 times) to generate ensembles of outputs, where each output is based on independent random samples from the defined parameter distributions. Running 10,000 ensembles for a basin with 30 sub-divisions over an 80 year input climate time series involves repeating the full set of model algorithms some 288×10^6 times, not to mention the time taken to exchange data with the SPATSIM database tables. A second version of the uncertainty model also allows the precipitation inputs to the model to be considered with uncertainty and makes use of stochastically generated rainfall sequences rather than a single fixed time series. Typically, the model is run with 500 stochastic rainfall sequences (for each spatial sub-division or catchment within the basin), in which case the number of parameter samples is limited to 500, giving a total number of 250,000 ensembles or 72×10^8 operations of the model algorithms. Fig. 2 illustrates the software configurations of the Delphi (SPATSIM) versions of the two models. These configurations were designed for sequential execution, but they can be parallelised by executing the functions within the ensemble loop on different threads with different input parameters, as is illustrated in Fig. 3.

The complexity of each model run and the design of the program can result in an undesirable model runtime of several hours on a modern CPU, even for the 10,000 ensembles (i.e., without stochastic rainfall inputs). An application of the stochastic rainfall version of the model to the Caledon River basin with 31 sub-catchments in southern Africa takes approximately 45 h to complete. While these model runs would not normally be repeated many times, as would be the case with a purely manual parameter search and calibration approach, it is sometimes useful to run the uncertainty model several times to explore the effects of