Environmental Modelling & Software 39 (2013) 295–303

Contents lists available at SciVerse ScienceDirect

Environmental Modelling & Software

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

High-performance computing tools for the integrated assessment and modelling of social–ecological systems[☆,☆☆]

Brett A. Bryan[*]

CSIRO Ecosystem Sciences, Waite Campus, Urrbrae, SA 5064, Australia

A R T I C L E I N F O

Article history:
Received 12 July 2011
Received in revised form
17 January 2012
Accepted 13 February 2012
Available online 10 March 2012

Keywords:
Graphics processing unit (GPU)
Parallel programming
Multi-core
Cluster
Grid
GIS
Environmental
Concurrency
Global challenges

A B S T R A C T

Integrated spatio-temporal assessment and modelling of complex social–ecological systems is required to address global environmental challenges. However, the computational demands of this modelling are unlikely to be met by traditional Geographic Information System (GIS) tools anytime soon. I evaluated the potential of a range of high-performance computing (HPC) hardware and software tools to overcome these computational barriers. Performance advantages were quantified using a synthetic model. Four tests were compared, using: a) an Arc Macro Language (AML) GIS script on a single central processing unit (CPU); b) Python/NumPy on 1–256 CPU cores; c) Python/NumPy on 1–64 graphics processing units (GPUs) with high-level PyCUDA abstraction (GPUArray); and d) Python/NumPy on 1–64 GPUs with low-level PyCUDA abstraction (ElementwiseKernel). The GIS implementation effectively took 15.5 weeks to run. Python/NumPy on a single CPU core led to a speed-up of 59× compared to the GIS. On a single GPU, speed-ups of 1473× were achieved using GPUArray and 4881× using ElementwiseKernel. Parallel processing led to further performance enhancements. At best, the ElementwiseKernel module in parallel over 64 GPUs achieved a speed-up of 63,643×. Open source tools such as Python applied across a spectrum of HPC resources offer transformational and accessible performance improvements for integrated assessment and modelling. By reducing the computational barrier, HPC can lead to a step change in modelling sophistication, including the better representation of uncertainty, and perhaps even new modelling paradigms. However, migration to new hardware and software environments also has significant costs. Costs and benefits of HPC are discussed and code tools are provided to help others migrate to HPC and transform our ability to address global challenges through integrated assessment and modelling.

1. Introduction

Addressing global environmental challenges (e.g. climate change, food and energy security, natural resource management, and biodiversity conservation) operating within complex social–ecological systems demands the *integrated* assessment and modelling of biophysical, ecological, economic, and social information (Costanza, 1996; Parker et al., 2002; Jakeman and Letcher, 2003; Kumar et al., 2006; Bryan et al., 2010a, 2011a; Reid et al., 2010). Many of these processes are heterogeneous over the landscape (Bryan, 2003) and may display spatial interconnections (e.g. topographic processes, species dispersal, supply chain analysis). Temporal dynamics may also be an important component in these processes (e.g. climate change, tree growth, cash flow) (Venevsky and Maksyutov, 2007). Data sets are often very large due to the increasing need to model processes over large extents (e.g. continental and global scale) whilst maintaining high enough spatial and temporal resolution (e.g. landscape scale, daily time-steps) to adequately capture the relevant dynamics (Harris, 2002). Further, uncertainty and sensitivity are an inherent part of these social–ecological systems which policy-makers need to understand in order to make robust decisions (Rotmans and van Asselt, 2001; Kooistra et al., 2005; Bryan, 2010; Cheviron et al., 2010; Lilburne and North, 2010). A common way to quantify sensitivity and uncertainty is by undertaking multiple model simulations using varying input parameter values (Lilburne and Tarantola, 2009). In concert, these characteristics of the integrated

Abbreviations: HPC, high-performance computing; CPU, central processing unit; GPU, graphics processing unit; NPV, net present value; AML, arc macro language; GIS, geographic information system.

☆ Computer codes are available in machine-readable form on request from the author.

☆☆ Thematic Issue on the Future of Integrated Modeling Science and Technology.

* Tel.: +61 8 8303 8581; fax: +61 8 8303 8582.
E-mail address: brett.bryan@csiro.au.

1364-8152/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.envsoft.2012.02.006

assessment and modelling of social–ecological systems have placed unprecedented demands upon both computer hardware and software. High-performance computing tools are required that enable us to tackle research problems of the scale and level of technological sophistication required to address global challenges (Openshaw, 1995; Openshaw and Turton, 2000; Armstrong et al., 2005; Yang and Raskin, 2009; Wang, 2010).

For the past few years, CPU clock speeds have plateaued at around 3 GHz due to physical limitations such as heat dissipation. To get around this barrier, chip designers have increased the number of cores on each chip (Herlihy and Shavit, 2008). Following Moore's law, the number of cores is now increasing exponentially (Sutter, 2005). This increase in parallelism is also reflected in the development of the GPU. With hundreds of cores on a single chip, GPUs have been developed primarily for the parallel rendering of computer graphics. The development of interfaces such as NVIDIA's Compute Unified Device Architecture (CUDA) and the Open Computing Language (OpenCL) have made general purpose GPU processing possible for scientific applications. In addition, the accessibility of massively parallel HPC resources has increased (Wang et al., 2005). High-speed networks have enabled the connectivity of multiple compute nodes, each containing multi-core CPUs and GPUs in server, cluster, and grid facilities (Jeffery, 2007; Herlihy and Shavit, 2008).

However, in order to utilise the performance advantages offered by HPC, models must be specifically written such that processes are executed concurrently across multiple cores and nodes (Rauber and Rünger, 2010). Two of the most common ways of achieving this in scientific programming are *data parallel* (where typically each processor executes the full program, but only on a share of the data) and *task parallel* (where typically each processor executes a share of the program on the full data set) (Rauber and Rünger, 2010). Whilst some integrated models may require communication between processes (e.g. agent-based models), many do not. The latter are termed *embarrassingly parallel*. Integrated assessment and modelling is often ideally suited to parallelization. This can be achieved either through data parallel methods such as tiling, distributing, and processing geographic data on multiple processors; or through task parallel methods such as processing a subset of simulations on multiple processors (Hawick et al., 2003; Armstrong et al., 2005).

In the past, spatio-temporal analysis has been typically undertaken in raster-based GIS and image processing software packages such as ArcGIS, GRASS, Imagine, ENVI, and others (e.g. Crossman and Bryan, 2009). These packages have been very effective for manipulating, managing, and analysing spatio-temporal data, and summarising and presenting results, particularly through high-quality cartographic outputs. They are also able to handle large data sets. However, the ability to implement integrated models of complex social–ecological systems in these packages is limited by the reliance on serial processing on a single CPU core and heavy disk input/output (I/O) transactions (Vokorokos et al., 2004). With few exceptions, traditional GIS and image processing software has lagged behind computer hardware trends towards increased parallelization and larger memory, thereby curtailing the progress of integrated assessment and modelling of social–ecological systems. Whilst proprietary packages such as MathWorks' Matlab and Wolfram's Mathematica have recently included the ability to use multiple CPUs and GPUs in parallel, these packages are not widely used by the spatial modelling community. Further, the deployment of these packages in large scale cluster environments can add a significant cost to research projects. The capacity of open source packages such as R (R Development Core Team, 2011) and Python (van Rossum and Python community, 2010) for performing spatio-temporal modelling and analysis has also progressed substantially during recent years. Additionally, both of these tools have packages that can use heterogeneous platforms featuring both CPUs and GPUs and spanning workstations through large compute clusters (e.g. Schmidberger et al., 2009).

The potential of HPC resources in spatio-temporal modelling of social–ecological systems was recognised early on Costanza and Maxwell (1991), Openshaw (1995), Clematis et al. (1996), Turton and Openshaw (1998), Openshaw and Abrahart (2000). Parallel computing on HPC resources has been used to analyse several aspects of social–ecological systems such as route optimisation (Lanthier et al., 2003), natural hazard simulation (Xie et al., 2010), atmospheric simulation (Wang et al., 2005), land surface modelling (Kumar et al., 2006), water resources management (Sulis, 2009; Kalyanapu et al., 2011), and urban simulation (Li et al., 2010). Several of these studies report performance improvements of orders of magnitude from running models on HPC resources. However, despite the substantial performance gains on offer, the use of HPC in the integrated assessment and modelling of social–ecological systems has not been widespread.

With the increased availability of both software and hardware for HPC, it is timely to rethink how we approach the integrated assessment and modelling of complex social–ecological systems. A critical and strategic evaluation of HPC software and hardware options is presented here to help practitioners thinking of migrating to these systems, including the actual code tools. I evaluated the ability of open source Python-based tools to undertake spatio-temporal modelling and analysis on HPC resources (a 128-node hybrid CPU/GPU supercomputer cluster). I developed an illustrative spatial model of economic returns to agriculture across a hypothetical heterogeneous landscape of 100 million grid cells (roughly equivalent to modelling the whole of Australia at a spatial resolution of 250 m). Yields, costs and prices were varied annually with climatic and market forces. Parameters for these are drawn from probability density functions. Net economic returns were calculated annually and discounted to net present value (NPV) terms. The NPV of agricultural returns was modelled over a time period of 70 years and this was repeated for 1000 Monte Carlo iterations. Four tests were compared, using: a) an Arc Macro Language (AML) GIS script on a single CPU; b) Python/NumPy on 1–256 CPU cores; c) Python/NumPy on 1–64 GPUs with high-level PyCUDA abstraction (GPUArray); and d) Python/NumPy on 1–64 GPUs with low-level PyCUDA abstraction (ElementwiseKernel). The advantages and disadvantages of this approach to integrated assessment and modelling are discussed. My hope is that this technology can provide a much-needed boost in our capability to address global environmental challenges.

2. Methods

2.1. Strategic evaluation

There are a range of tools available for parallel processing on CPUs and GPUs. With a focus on open source tools, excellent array processing capability exists with both Python (van Rossum and Python community, 2010) in the *NumPy* package (Jones et al., 2001), and R (R Development Core Team, 2011). Within R, many packages offer some ability for parallel CPU processing (Schmidberger et al., 2009) although I had limited success with several of these. *The rmpi* package (Yu, 2010) was an exception and enabled parallel processing on large scale implementations via Message Passing Interface (MPI). Within Python there are also several options for parallel processing (e.g. *multiprocessing*, *subprocess*) which can work well in smaller implementations (e.g. on multi-core machines). However, *IPython* (Perez and Granger, 2007) was found to scale to highly parallel processing on heterogeneous clusters. R has a few packages (e.g. *gputools* Buckner et al., 2010) which provide specific routines for GPUs. For Python, the PyCUDA library provides full access to the CUDA API and abstractions making the programming of common tasks easier. Other libraries (e.g. *PyOpenCL* (Klöckner et al., 2011), *Theano* (Bergstra et al., 2010)) can also be used for GPU processing through Python. Excellent graphical and mapping ability is available within both Python (*Matplotlib*; Hunter, 2007) and R (e.g. *ggplot2*; Wickham, 2009). Whilst R has a more comprehensive suite of analytical tools, Python tends to be faster, better able to handle large data sets, and provides more