Computer Physics Communications 185 (2014) 1265–1272

Contents lists available at ScienceDirect

Computer Physics Communications

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

Fast finite difference Poisson solvers on heterogeneous architectures

Pedro Valero-Lara [a,c,*], Alfredo Pinelli [b], Manuel Prieto-Matias [c]

[a] Unidad de Modelización y Simulación Numérica, CIEMAT, Madrid, Spain

[b] School of Engineering and Mathematical Sciences, City University London, UK

[c] Departamento de Arquitectura de Computadores, Facultad de Informática, Universidad Complutense de Madrid (UCM), Spain

ARTICLE INFO

Article history:
Received 7 February 2013
Received in revised form
20 December 2013
Accepted 27 December 2013
Available online 3 January 2014

Keywords:
Fast finite difference Poisson solvers
Parallel computing
CPU–GPU heterogeneous architectures

ABSTRACT

In this paper we propose and evaluate a set of new strategies for the solution of three dimensional separable elliptic problems on CPU–GPU platforms. The numerical solution of the system of linear equations arising when discretizing those operators often represents the most time consuming part of larger simulation codes tackling a variety of physical situations. Incompressible fluid flows, electromagnetic problems, heat transfer and solid mechanic simulations are just a few examples of application areas that require efficient solution strategies for this class of problems. GPU computing has emerged as an attractive alternative to conventional CPUs for many scientific applications. High speedups over CPU implementations have been reported and this trend is expected to continue in the future with improved programming support and tighter CPU–GPU integration. These speedups by no means imply that CPU performance is no longer critical. The conventional CPU-control–GPU-compute pattern used in many applications wastes much of CPU's computational power. Our proposed parallel implementation of a classical cyclic reduction algorithm to tackle the large linear systems arising from the discretized form of the elliptic problem at hand, schedules computing on both the GPU and the CPUs in a cooperative way. The experimental result demonstrates the effectiveness of this approach.

1. Introduction

The era of single-threaded processors has come to an end due to the limitation of the CMOS technology and in response, most hardware manufacturers are designing and developing multicore processors and specialized hardware accelerators such as GPUs [1–3]. As a consequence, applications can only improve their performance if they are able to exploit the available parallelism of the new architectures.

In this paper we study the implementation of a fast solver based on a block cyclic reduction algorithm to tackle the linear systems that arise when discretizing a three dimensional separable elliptic problem with standard finite difference. A clear example of the importance of dealing efficiently with three dimensional elliptic systems is found in the numerical simulation of incompressible fluid flows. Indeed, the most time consuming part of almost any incompressible unsteady Navier Stokes solver (i.e., incompressible fluid dynamic simulation codes) is related to the solution of a *pressure Poisson* equation at each time step (see for instance [4]). The achievement of a satisfactory computational efficiency to tackle this class of elliptic partial differential equations is therefore a key issue when simulating unsteady fluid flow processes (turbulent flows for instance).

Other authors have addressed topics which are somehow related to the present contribution. C. P. Stone et al. [5] analyze the performance of a block tridiagonal benchmark on GPUs. This is the first known implementation of a block tridiagonal solver in CUDA but the pattern of the block matrices they analyzed differ from our target problem. The sub-matrix element rank (m) was assumed to be small ($m = 5$). In our case both m and the arithmetic intensity of problem are higher.

For distributed multicore clusters, the BCYCLIC algorithm developed by Hirshman et al. [6] is able to solve linear problems with dense tridiagonal blocks. Our target algorithm, the BLKTRI code [7] is not well-suited for dense blocks but it is the most popular approach for solving block tridiagonal matrices which arise from separable elliptic partial differential equations.

Many authors have studied the implementation of scalar tridiagonal solver on GPUs [8–13]. D. Goddeke et al. [8] proposed an efficient implementation of the Cyclic Reduction (CR) algorithm, which is used as a line smoother in a multigrid solver. Yao Zhang et al. [9] proposed some hybrid algorithms that combine CR with other tridiagonal solvers such as Parallel Cyclic Reduction (PCR) or Recursive Doubling (RD). More recently, H. Kim et al. [12] have analyzed other hybrid algorithms and found that a combination of PCR and Thomas gave the best overall performance.

* Corresponding author at: Unidad de Modelización y Simulación Numérica, CIEMAT, Madrid, Spain.

E-mail addresses: pedro.valero.lara@gmail.com, pedro.valero@ciemat.es (P. Valero-Lara), Alfredo.Pinelli.1@city.ac.uk (A. Pinelli), mpmatias@dacya.ucm.es (M. Prieto-Matias).

0010-4655/$ – see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cpc.2013.12.026

The rest of this paper is structured as follows. Section 2 introduces the extended block cyclic reduction algorithm used by the BLKTRI solver. Section 3 gives a brief description of the standard algorithms for solving scalar tridiagonal systems. In Section 4 we detail the mapping of the BLKTRI solver on multicore and GPUs and analyze their performance and then in Section 5 we extend our discussion to 3D problems. Finally, Section 6 concludes summarizing the most relevant contributions.

2. Three dimensional elliptic systems

In this section, we explain the strategy followed to solve a classical 3D Poisson equation:

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}=f(x,y,z)$$

defined on a Cartesian domain Ω with prescribed conditions on its boundary $\partial\Omega$.

Discretizing the domain using a Cartesian mesh uniform along each direction, for each (i,j,k) interior node we obtain:

$$\delta_x^2(i,j,k)+\delta_y^2(i,j,k)+\delta_z^2(i,j,k)=f_{i,j,k} \tag{1}$$

where

$$\delta_x^2(i,j,k)=\left(u_{i-1,j,k}-2u_{i,j,k}+u_{i+1,j,k}\right)/\Delta x^2$$
$$\delta_y^2(i,j,k)=\left(u_{i,j-1,k}-2u_{i,j,k}+u_{i,j+1,k}\right)/\Delta y^2$$
$$\delta_z^2(i,j,k)=\left(u_{i,j,k-1}-2u_{i,j,k}+u_{i,j,k+1}\right)/\Delta z^2$$

are the finite difference centered approximations to the second derivatives along each direction. The boundary conditions that we will consider are either of Dirichlet or Neumann type on the surfaces normal to the y and z directions and periodic in the x one. The periodic condition applied in one of the directions allows to uncouple the 3D problem into a set of several independent 2D problems (Fig. 1) using a discrete Fourier transform. Hereafter we will briefly explain how the decoupling process takes place. Let N be the number of equispaced nodes in the x direction that cover the interval $(0,2\pi)$. We expand the unknown function $u(x,y,z)$ and $f(x,y,z)$ in Fourier series as:

$$u_{n,j,k}=\frac{1}{N}\sum_{l=1}^{N}\hat{u}_{l,j,k}e^{-i\alpha(n-1)}\quad\text{with }\alpha=\frac{2\pi(l-1)}{N} \tag{2}$$

where $\hat{u}_{l,j,k}$ is the lth Fourier coefficient of the expansion. Next, the expansion is used in Eq. (1), obtaining the relationship:

$$\frac{1}{N}\sum_{l=1}^{N}e^{-i\alpha(n-1)}\left\{\frac{\hat{u}_{l,j,k}}{\Delta x^2}(e^{-i\alpha}-2+e^{i\alpha})+\delta_y^2\hat{u}_{l,j,k}+\delta_z^2\hat{u}_{l,j,k}\right\}$$
$$=\frac{1}{N}\sum_{l=1}^{N}\hat{F}_{l,j,k}e^{-i\alpha n}. \tag{3}$$

Eq. (3) is equivalent to the set of N equations ($l=1\ldots N$):

$$\frac{\hat{u}_{l,j,k}(2\cos(\alpha)-2)}{\Delta x^2}+\frac{\hat{u}_{l,j+1,k}-2\hat{u}_{l,j,k}+\hat{u}_{l,j-1,k}}{\Delta y^2}$$
$$+\frac{\hat{u}_{l,j,k+1}-2\hat{u}_{l,j,k}+\hat{u}_{l,j,k-1}}{\Delta z^2}=\hat{F}_{l,j,k} \tag{4}$$

having used the identity $e^{i\alpha}+e^{-i\alpha}=2\cos(\alpha)$. In short (4) reads as:

$$\frac{\hat{u}_{l,j+1,k}+\hat{u}_{l,j-1,k}}{\Delta y^2}+\frac{\hat{u}_{l,j,k+1}+\hat{u}_{l,j,k-1}}{\Delta z^2}+\beta_l\hat{u}_{l,j,k}=\hat{F}_{l,j,k},$$
$$l=1\ldots N \tag{5}$$

with $\beta_l/2=\cos(\alpha)-1/\Delta x^2-1/\Delta y^2-1\Delta z^2$. Thus, by considering the Fourier transform (direct FFT) of F one obtains a set of N, 2D independent problems having as unknowns the Fourier coefficients $\hat{u}_{l,j,k}$, $l=1\ldots N$. Each independent problem concerns the solution of a linear system of equations which coefficient matrix is block tridiagonal. Of course, each one of these linear systems can now be solved in a distributed fashion, in parallel. Once the solution is obtained in Fourier space a backward FFT can be used to recast the solution in physical space. Fig. 1 provides an algorithmical sketch of the method.

To deal with each decoupled 2D problem, we have chosen a direct method based on a block cyclic reduction algorithm. As shown above, the whole method provides for a blend of coarse and fine-grain parallelism that can be exploited when mapped on heterogeneous platforms.

2.1. Extended block cyclic reduction

In this subsection we briefly summarize a classical direct method for the discrete solution of separable elliptic equations based on a block cyclic reduction algorithm [7]. This method is commonly used when tackling the solution of a linear system of equations arising from the second order centered finite difference discretization of 2D separable elliptic equations. From the standpoint of computational complexity (speed and storage), for a $m\times n$ grid, its operation count is proportional to $mn\log_2 n$, and the storage requirements are minimal, since the solution is returned in the storage occupied by the right side of the equation (i.e., $m\times n$ locations are required). More in details, consider the 2D separable elliptic equation having $u(x,y)$ as unknown field (Poisson equation is a particular case of what follows):

$$\frac{\partial}{\partial x}\left(a(x)\frac{\partial u}{\partial x}\right)+b(x)\frac{\partial u}{\partial x}+c(x)u$$
$$+\frac{\partial}{\partial y}\left(d(y)\frac{\partial u}{\partial y}\right)+e(y)\frac{\partial u}{\partial y}+f(y)u=g(x,y). \tag{6}$$

If we discretize (6) with given Dirichlet or Neumann boundary conditions assigned on the edges of a square, using the usual five-point scheme with the discrete variables ordered in a lexicographic fashion, we obtain a linear system of $m\times n$ equations (having m nodes in the x direction and n in y one): $\mathbf{A}\vec{\mathbf{u}}=\vec{\mathbf{g}}$, where $\mathbf{A}$ is a block tridiagonal matrix:

$$\mathbf{A}=\begin{bmatrix}B_1 & C_1 & & & & 0\\ A_2 & B_2 & C_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & A_{n-1} & B_{n-1} & C_{n-1}\\ & & & & A_n & B_n\end{bmatrix}$$

and the vectors $\vec{u}$ and $\vec{g}$ are consistently split as a set of sub-vectors $\vec{u}_j$ and $\vec{g}_j$, $j=1\ldots n$, of length m each (i.e., the solution along the jth domain row):

$$\mathbf{u}=\left[\vec{\mathbf{u}}_1,\ \vec{\mathbf{u}}_2,\ldots,\vec{\mathbf{u}}_n\right]^T$$
$$\mathbf{g}=\left[\vec{\mathbf{y}}_1,\ \vec{\mathbf{y}}_2,\ldots,\vec{\mathbf{y}}_n\right]^T.$$

There is no restriction on m; however, cyclic reduction algorithms require $n=2^k$, with large values of k for optimal performances. The blocks $\mathbf{A_i}$, $\mathbf{B_i}$ and $\mathbf{C_i}$ are $m\times m$ square matrices. In particular, the *BLKTRI* algorithm requires them to be of the form:

$$A_i=a_iI \tag{7}$$
$$B_i=B+b_iI \tag{8}$$
$$C_i=c_iI \tag{9}$$

where a_i, b_i and c_i are scalars. Having used a standard five point stencil for the discretization of (6), the matrix B is of tridiagonal pattern. The solution is obtained using an *extended cyclic reduction*