SAND2016-0607C

IMPLEMENTATION OF EXASCALE ASYNCHRONOUS SOLVERS FOR PDEs

Konduri Aditya[1,2], Diego A. Donzis *[1], and Lawerence Rauchweger[3]

[1]*Department of Aerospace Engineering, Texas A&M University, Texas, United States of America*
[2]*Combustion Research Facility, Sandia National Laboratories, California, United States of America* †
[3]*Department of Computer Science and Engineering, Texas A&M University, Texas, United States of America*

Summary To overcome the challenges likely to be present in exascale systems we recently developed novel asynchrony-tolerant numerical schemes for partial differential equations which relax the need for synchronization between processing elements at a mathematical level. The advantages of such schemes can only be realized if their implementation also aims at alleviating the limiting factors at exascale in terms of hardware, software, and programing frameworks. We present two such implementations with different degrees of asynchrony and their corresponding tradeoffs between numerical accuracy and performance. Analysis and numerical experiments demonstrate their respective potential to increase scalability of solvers in future exascale systems.

INTRODUCTION

In the view of the challenges in scalability of solvers likely to be present on future exascale systems, we recently proposed [2] an asynchronous computing method for partial differential equations (PDEs) based on finite differences. This method relaxes data synchronization between processing elements (PEs) at a mathematical level, which potentially can reduce the communication time and improve scalability. An important conclusion from our work is that the numerical properties of those schemes also depend on the characteristics of the hardware and software on a particular system in addition to standard numerical and physical parameters. While the mathematical foundation of asynchronous methods has been established in [2, 1], a number of alternatives open up in terms of implementation of solvers for simulations at extreme scales on real exascale systems. Different choices in implementation can have an impact on, among others, communication patterns, contention on subsystems, and the potential degree of computations-communication overlap. In this work, we present two implementations which result in two broad families of schemes to highlight their capabilities in terms of numerical accuracy and computational performance. As we show, in both cases there is a trade-off between performance and accuracy. The focus on the comparison between implementations will shed light on how to quantitatively strike the right balance in realistic simulations.

CONCEPT

Consider the one-dimensional linear advection-diffusion equation, given by $\partial u/\partial t + c\partial u/\partial x = \alpha\partial^2 u/\partial x^2$, where c and α are constants, discretized with an Euler scheme in time and a second order central difference in space: algebraic equation:

$$\frac{u_i^{n+1}-u_i^n}{\Delta t} + c\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} = \alpha\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} + \mathcal{O}(\Delta x^2, \Delta t) \tag{1}$$

The solution is evolved in time for given initial and boundary conditions. When the domain is decomposed into multiple PEs, the computation of spatial derivatives near PE boundaries need u values from neighboring PEs. Thus, at each time step, these computations cannot proceed unless communications between PEs is completed, thus enforcing synchronization.

This synchronization, however, can be relaxed by allowing schemes to compute spatial derivatives using function values that are not current but from delayed time levels. This modifies the nature of the finite difference equation, affecting also its numerical properties. The mathematical feasibility of this method has been studied in [2] where we found that though standard schemes remain stable under asynchrony, their accuracy is significantly affected. In [1] we derived families of asynchrony-tolerant (AT) schemes of arbitrary order under asynchronous conditions. An example of a second-order AT scheme is

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1}^n - 2u_i^n + (k+1)u_{i-1}^{n-k} - ku_{i-1}^{n-k-1}}{\Delta x^2}, \tag{2}$$

where k is the delay in the latest available time level at the grid point $i-1$, which can be set deterministically or randomly depending on the implementation. This difference is the basis for selecting the implementations presented here. The maximum delay can be restricted to L time levels (i.e. $k \in \{0, 1, \ldots, L-1\}$), providing a knob to control error. We define $\mathcal{T}$ as the number of asynchronous time levels that a particular scheme needs, which in the case of Eq. (2) is $\mathcal{T} = 2$, corresponding to the time levels $\{n-k, n-k-1\}$. The average error due to asynchrony can be then shown [1] to be proportional to $(P/N)\Delta x^a \sum_{m=1}^{\mathcal{T}} \gamma_m \overline{k^m}$, where γ_m are some constants, and P, N and a are the number of PEs, the number of grid points and the order of accuracy of the AT scheme, respectively. An overbar is an ensemble average. This result clearly shows that the accuracy of asynchronous schemes at exascale depends not only on the numerical scheme but also on architectural details of the system (through statistics of the delays $\overline{k^m}$) as well as the manner in which problems are scaled up (through P/N).

*Corresponding author. Email: donzis@tamu.com
†Current affliation of KA.

IMPLEMENTATIONS OF NUMERICAL SCHEMES WITH DELAYS

We present two implementations with different communication algorithms. In the first one, which will be referred as communication minimizing (CM) implementation, communication of data with synchronization is not carried out at every time step in a simulation, but after a given number of time steps. This reduces the number of messages that are sent during a simulation to improve scalability. Specifically, communications are done during $\mathcal{T}$ time levels after which computations continue without communications for $L - \mathcal{T}$ time levels. The process then repeats. Note that the delay k varies in a deterministic manner in this implementation. For a given physical problem represented by a PDE, error control is based on the particular AT scheme used and the value of L. They also have an impact on size, frequency and number of messages during communications. We show how one can predictably trade performance and accuracy with these parameters.

The second implementation is based on asynchronous communications, in which communication between PEs is initiated at every time step but no explicit synchronization is enforced as long as $k < L - 1$, where L is set based on accuracy requirements. We refer to this as the weakly synchronous (WS) implementation. The scalability of solvers is then improved by overlapping computations and communications which is possible due to the removal of explicit synchronizations at every time step. In such an implementation, message arrivals are in general random and so is the delay k.

RESULTS AND CONCLUSIONS

To evaluate the performance of our asynchronous methods we have conducted simulations of the advection-diffusion equations, which represent a widely used case for a PDE with realistic physical content. As a reference case, we choose the standard synchronous scheme in Eq. (1). The results show that the CM and WS implementations give rise to different moments of the delay k. This has a first order effect on the error incurred. As we show in this work, in both implementations we can control the error using L though the effect is different. An example of the numerical convergence of both implementations is shown in Fig. 1(a) as grid resolution is increased. When compared to the standard synchronous case (black line), the error incurred by using a standard (non-AT) scheme asynchronously is much greater and converges at a lower rate (red line). However, this effect is relieved by using the AT scheme in Eq. (2), as seen in the figure (green line).

Performance at extreme scales will also depend on the particular scheme, the statistics of the delay, and L at which synchronization is enforced. We show how these elements combine in each implementation. An example of the computational performance is shown in the strong scaling graph in Fig. 1(b). The computational effort per PE used in these simulations is similar to the expected values at exascale. We see that the standard synchronous case scales poorly as we increase the PE count, due to the dominance of communication time. When the communication synchronization is relaxed using e.g. WS implementation, a nearly ideal scaling is obtained. A similar behaviour is also observed in the case of CM implementation.

In conclusion, this work combines recent advances in the mathematical understanding of AT schemes with elements on the actual implementation on real architectures and its consequences in accuracy and performance. We demonstrate how specific implementations of AT schemes can provide a path towards true exascale simulations of problems governed by PDEs. We also note that since the complexity in writing asynchronous exascale codes (especially in high dimensional space and with complex geometries) is expected to be extremely high, a framework that provide mechanisms in which asynchronous schemes can be expressed and analyzed efficiently [3] will be extremely valuable.

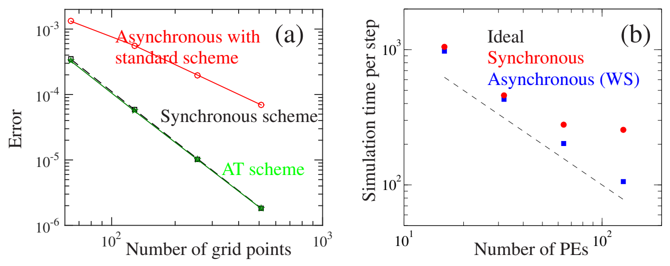

Figure 1: (a) Convergence of error with increase in grid resolution. (b) Strong scaling graph.

References

[1] K. Aditya. *Highly scalable asynchronous computing method for partial differential equations: a path towards Exascale*. PhD thesis, TAMU, 2015.
[2] D. A. Donzis and K. Aditya. Asynchronous finite-difference schemes for partial differential equations. *J. Comp. Phys.*, 274(0):370 – 392, 2014.
[3] M. Zandifar et al. Composing algorithmic skeletons to express high-performance scientific applications. In *Proceedings of ICS*, pages 415–424, 2015.