MAP55640 - Comparative Analysis of Hybrid-Parallel Finite Difference Methods on Multicore Systems and Physics-Informed Neural Networks on CUDA Systems

LI YIHAI, Mathematics Institute High Performance Computing, Ireland
MIKE PEARDON*, Mathematics Institute High Performance Computing, Ireland

This is the abstract of this project.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Additional Key Words and Phrases: Keywords

1 INTRODUCTION

Numerical methods of solving partial differential equations (PDE) have demonstrate far better performance than many other methods such as finite difference methods (FDM) [**<empty citation>**], finite element methods (FEM) [**<empty citation>**], Lattice Boltzmann Method (LBM) [**<empty citation>**] and Monte Carlo Method (MC) [**<empty citation>**]. In recent years, researchers in the field of deep learning have mainly focused on how to develop more powerful system architectures and learning methods such as convolution neural networks (CNNs) [**<empty citation>**], Transformers [**<empty citation>**] and Perceivers [**<empty citation>**] . In addition, more researchers have tried to develop more powerful models specifically for numerical simulations. Despite of the relentless progress, modeling and predicting the evolution of nonlinear multiscale systems which has inhomogeneous cascades-scales by using classical analytical or computational tools inevitably encounts severe challanges and comes with prohibitive cost and multiple sources of uncertainty.

This project focuses on the promotions on performance gained from the parallel compute systems, in general, the FDMs and Neural Networks (NNs) are evaluated. Moreover, it prompted series Message Passing and Shared

*Supervisor of this Final Project

Authors' addresses: Li Yihai, liy35@tcd.ie, Mathematics Institute and High Performance Computing, Dublin, Ireland; Mike Peardon, mjp@maths.tcd.ie, Mathematics Institute and High Performance Computing, Dublin, Ireland.

2024. Manuscript submitted to ACM

Memory hybrid parallel strategies using Message Passing Interface (MPI) [**MPI**] and Open Multi-processing (OpenMP) [**OpenMP**].

2 RELATED WORK

To gain well quality solution of various types of PDEs is prohibitive and notoriously challanging. The number of methods avaliable to determine canonical PDEs is limited as well, includes separation of variables, superposition, product solution methods, Fourier transforms, Laplace transforms and perturbation methods, among a few others. Even though there methods are exclusively well-performed on constrained conditions, such as regular shaped geometry domain, constant coefficients, well-symmetric conditions and many others. These limits strongly constrained the range of applicability of numerical techniques for solving PDEs, rendering them nearly irrelevant for solving problems pratically.

General, the methods of determining numerical solutions of PDEs can be broadly classified into two types: deterministic and stochastic. The mostly widely used stochastic method for solving PDEs is Monte Carlo Method [**Monte Carlo Method**] which is a popular method in solving PDEs in higher dimension space with notable complexity.

2.1 Finite Difference Method

The Finithe Difference Method(FDM) is based on the numerical approximation method in calculus of finite differences. The motivation is quiet straightforward which is approximating solutions by finding values satisfied PDEs on a set of presctibed interconnected points within the domain of it. Those points are which referd as nodes, and the set of nodes are so called as a grid of mesh. A notable way to approximate derivatives are using Taylor Series expansions. Taking 2 dimension Possion Equation as instance, assuming the investigated value as, φ,

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = f(x, y) \tag{1}$$

The total amount of nodes is denoted with $N = 15$, which gives the numerical equation which governing equation 1 shown in equation 2 and nodes layout as shown in the figure 1

$$\frac{\partial^2 \varphi_i}{\partial x_i^2} + \frac{\partial^2 \varphi_i}{\partial y_i^2} = f(x_i, y_i) = f_i, \quad i = 1, 2, \ldots, 15 \tag{2}$$

In this case, we only need to find the value of internal nodes which i is ranging from 1 to 10. Next is aimming to solve this linder system 2.

2.2 Physics Informed Neural Networks

With the explosive growth of avaliable data and computing resouces, recent advances in machine learning and data analytics have yieled good results across science discipline, including Convolutional Neural Networks (CNNs) [**CNN**] for image recoginition, Generative Pre-trained Transformer (GPT) [**GPT**] for natual language processing and Physics Informed Neural Networks (PINNs) [**PINN**] for handling science problems with high complexity. PINNs is a type of machine learning model makes full use of the benefits from Auto-differentiation (AD) [**AD**] which led to the emergence of a subject called Matrix Calculus [**Matrix_Calculus**]. Considering the parametrized and nonlinear PDEs of the general form [E.q. 3] of function $u(t, x)$

$$u_t + \mathcal{N}[u; \lambda] = 0 \tag{3}$$

The $\mathcal{N}[\cdot; \lambda]$ is a nonlinear operator which parametrized by λ. This setup includes common PDEs problems like heat equation, and black-stokz equation and others. In this case, we setup a neural network $NN[t, x; \theta]$ which has trainable weights θ and takes t and x as inputs, outputs with the predicting value $\hat{u}(t, x)$. In the training process, the next step is

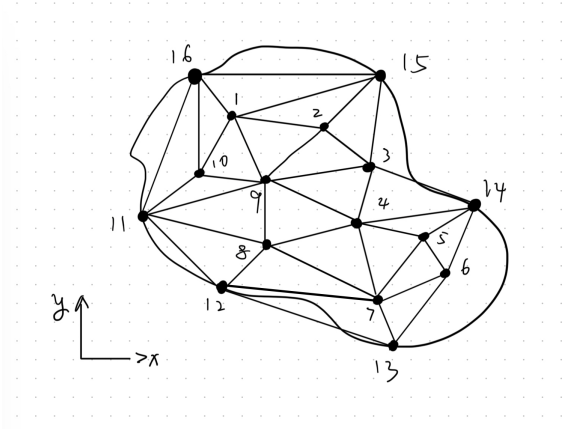

Fig. 1. The Schematic Representa of of a 2D Computiational Domain and Grid. The nodes are used for the FDM by solid circles. Nodes $11-15$ denote boundary nodes, while nodes $1-10$ denote internal nodes.

calculating the necessary derivatives of u with the respect to t and x. The value of loss function is a combination of the metrics of how well does these predictions fit the given conditions and fit the natural law [Fig. 2].

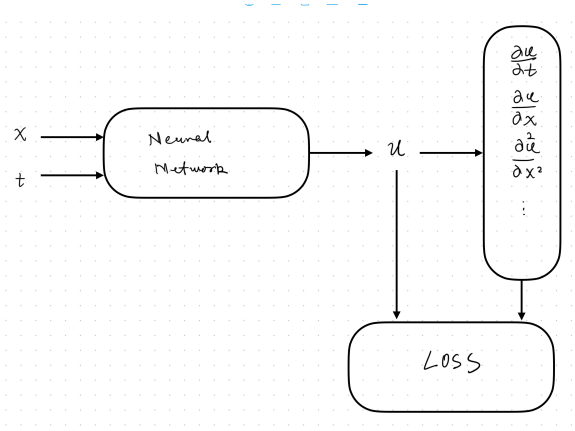

Fig. 2. The Schematic Representa of a structure of PINN.

2.3 Finite Difference Time Domain Method

As described previously in the section 2.1, FDM could solve the PDEs in its original form where Finite Element and Finite Volume Methods gained results by solving modified form such as an integral form of the governing equation. Though the latter methods are commonly get better results or less computational hungery, the FDM has many descendants, for instance the Finite Difference Time Domain Method (FDTD) where it still finds prolific usage are computational heat equation and computational electromagnetics (Maxwell's equations [**Maxwell_equations**]). Assuming the operator

$\mathcal{N}[\cdot;\lambda]$ is set to ∇ where it makes E.q. 3 become to heat equation 4.

$$\frac{\partial u}{\partial t} - \lambda\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0 \tag{4}$$

Using the key idea of FDM, assuming the step size in spatio-time space are Δx, Δy and Δt, we could have a series equations which have form [E.q. 5].

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = \lambda\left(\frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{\Delta x^2} + \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j+1}^{n}}{\Delta y^2}\right) \tag{5}$$

when the time step size satisfies the Couran, Friedrichs, and Lewy condition (CFL[**CFL_limitation**]). We could get the strong results by iterating the equation 5, or more specifically using equation 6 to get the value u of next time stamp $n+1$ on nodes i, j.

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \frac{\lambda\Delta t}{\Delta x^2}\left(u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}\right) + \frac{\lambda\Delta t}{\Delta y^2}\left(u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j+1}^{n}\right) \tag{6}$$

also shown in the figure 3.

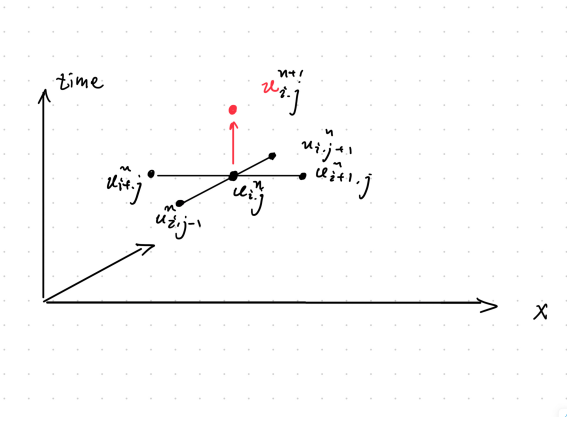

Fig. 3. The Schematic Representa the computational spatio-time domain of FDTD methods.

3 PROBLEM SETUPS

Due to the inherent limitations of current computing systems, obtaining sufficiently precise solutions is both computationally expensive and time-consuming. These challenges arise from the constraints imposed by the clock speed of computing units, as described by Moore's Law [**Moore_Law**], as well as the relatively low communication speeds between these units. While modern numerical methods have advanced to a level where they can produce satisfactory results within acceptable time frames across many research domains, the increasing scale of problems we aim to solve has driven the search for more cost-effective approaches. This has led to a growing interest in neural networks as a promising alternative.

In this project, I aim to evaluate the performance of the Finite-Difference Time-Domain (FDTD) method and the Physics-Informed Neural Network (PINN) model within parallelized computing environments by find the steady-state solution of PDEs. These two methodologies broadly represent the current approaches to handling PDEs, specifically CPU-based parallelization and GPU-based parallelization.

3.1 General Form

Starting with the general form of the PDEs, rather than the specific euqations, is because different equations give perform differently on the same compute system. To this end, consider the previously discussed form of PDEs shown in equation 3 which parametrized by number λ and an operator $\mathcal{N}[\cdot;\lambda]$. Moreover, we assume the variable x is a 2D or 3D spatio vector which is written in $\vec{x} \in \mathbb{R}^d$, $d = 2, 3$.

$$\begin{aligned} &\frac{\partial u}{\partial t}(t,\vec{x}) + \mathcal{N}[u(t,\vec{x});\lambda] = 0, && \vec{x} \in \Omega, t \in [0,+\infty) \\ &u(0,\vec{x}) = \varphi(\vec{x}), && \vec{x} \in \Omega \\ &u(t,\vec{x}) = g(t,\vec{x}), && \vec{x} \in \overline{\Omega}, t \in [0,+\infty) \end{aligned} \tag{7}$$

The domain of this PDE system is considered between 0 and 1, where is denoted with $\Omega = [0,1]^d$, $d = 2, 3$. To these setups, we have the general form of the PDEs we are going to investigated, shown in equations 7 The boundary condition shown in E.q. 7 is Dirichlet Condition known as first type boundary condition, where as the second type boundary condition (Von Neumman) [E.q. 8] gives the other form of $u(t,\vec{x})$ at the boundary $\overline{\Omega}$.

$$\frac{\partial u}{\partial \vec{x}} = g(t,\vec{x}), \quad \vec{x} \in \overline{\Omega}, \quad t \in [0,+\infty) \tag{8}$$

3.2 Computiational Topology

The computational topology is critically important when we are programming parallel PDEs solver softwares. Put the strongly speed-dependent data into the slow memory could make entire program slower.

Callan. The cluster we are using for this project is `Callan` [**Callan_TCD**] which has 2 CPUs per compute node, and each CPU has 32 cores with single thread. The Non-Uniform Memory Access (NUMA) nodes are layout as following Accessing the other NUMA node's memory reduces the bandwidth and also the latency, though the bandwidth is commonly high enough, the latency can increase by 30% to 400% [**NUMA_Latency_TCD**]. This latency becomes dangerous when writing shared memory parallel programs.

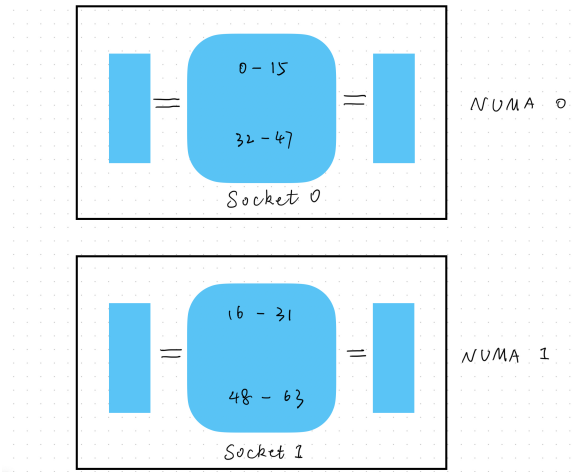

Fig. 4. NUMA topology of single node on Callan

3.3 Accuracy

When we tried to represent numbers using arithmetic in binary, decimal or hexadecimal, truncation always affects the precision of every number, or so called as round-off-error.

3.3.1 Round-off Error. In IEEE-754 [**IEEE_754**] standards, a 32-bit floating pointer number, single precision, obligatorily represented with 23-bit mantissa, 8-bit exponent and 1-bit for sign. Where as 64-bit floating number, double precision, also ubiquitous used, which has 11-bit exponent and 52-bit mantissa. After almost three decades development, not only single and double precisions (float32, float64) are ubiquitously in use, also more formats such as fp4, fp8, and fp16 etc. Both of them follows the simple form of exponent k, sign n and mantissa N. [**IEEE_754_p2_eq1**]

$$2^{k+1-N}n$$

Round-off errors are a manifestation of the fact that on a digital computer, which is unavoidable in numerical computations. In such case, the precision of the number depends on how many bytes are used to store single number. For instance, a float32 number provides $2^{-23} \approx 1.2 \times 10^{-7}$, and a double precision number gives $2^{-53} \approx 2.2 \times 10^{-16}$, such number is called machine ϵ which is the smallest number the machine can represent with given format.

In numerical methods I investigated, the FDTDs are conventionally using double precision number so that the programs can treat extreamly large and small numbers simultaneously in the same computation, without worring about the round-off errors. However, as mentioned, fp32 and fp16 are also popular use in scientific computing, especially in machine learning training process. While the lasted training GPUs are integrated compute accelarate unit for low precision floating numbers. [**NVIDIA_HB200_PAPER**].

3.3.2 Floating-point Arithmetic. The other type loss comes from the arithmetic operations on two numbers x, y. The standard model holds that

$$fl(x \text{ op } y) = (x \text{ op } y)(1 + \delta), \quad |\delta| < \epsilon \tag{9}$$

where the op stands for the four elementary operations: $+, -, \times, /$. [**Germund**, **NMSC**, **V1**, **P112**].

4 METHEDOLOGY

4.1 Distritization

To begin with descritizing the objects or the region we are going to evaluate via matrices. Without consdering more detail, the navie way for these transforming processes is to simply using the coordinates in $d = 2, 3$ dimension spaces and the values of function at the points to simplify the objects, which is a fine way to investigate objects with regular shapes such as Cube. For the FDTD, we are using a fine generated d-Cube which has shape $\{n_i\}_i^d$, including the boundary conditions, the cude has nodes $\Pi_i(n_i + 2)$ numbers. And it takes $4\Pi_i(n_i + 2)$ bytes for float32 or $8\Pi_i(n_i + 2)$ bytes for float64 to store in the memory. With such setup, for equally spaced nodes, we get

$$\Delta x_i = \frac{1}{n_i - 1} \tag{10}$$

Fig. 5. Figure Shows the FDM idea, If NEED

4.2 Finite Difference Methods

4.2.1 *Pure Message Passing Parallel.*

4.2.2 *Hybrid Parallel.*

4.3 Physics Informed Neural Networks

4.3.1 *CUDA parallel.*

4.3.2 *Hybrid Parallel.*

5 IMPLEMENTATION

5.1 Finite Difference Methods

5.1.1 *Pure Message Passing Parallel.*

5.1.2 *Hybrid Parallel.*

5.2 Physics Informed Neural Networks

5.2.1 *CUDA parallel.*

5.2.2 *Hybrid Parallel.*

5.3

6 EXPERIMENTS

7 CONCLUSION

8 ACKNOWLEDGEMENT