Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 32 No. 1 (2025)

Parallel Computing Techniques for Solving Large-Scale Partial Differential Equation

Mr. Vinayak Kishan Nirmale[1], Mrs. S Hemalatha[2], Dr. K. Santosh Reddy[3], Dr. Ashok Reddy Thontla[4], Dr. Pamarthi Sunitha[5], Dr. T. Lawanya[6], Mr. Divvela Surendra[7],

[1]Lecturer in Mathematics, Department of Polytechnic, MIT World Peace University, Pune, Maharashtra, India, Pincode: 411038. Email : vinayak.nirmale@mitwpu.edu.in

[2]Assistant Professor, Sir C R Reddy College of Engineering (Affiliated by JNTU Kakinada), Eluru, Andhra Pradesh, India, Pincode: 534007 Email id : hematrinadh25@gmail.com

[3]Assistant Professor of Mathematics, Vardhaman College of Engineering, Hyderabad, Telangana, India, Pincode: 501218, Email id: santureddyk@gmail.com

[4]Manager - Analytics, Department of IT, Cognizant Srihanuman Arcade, KPHB 9 phase, Hyderabad, India, Pincode: 500085 Email id: ashokthontla@gmail.com

[5]Associate Professor, Department of Electronics and Communication Engineering, Aditya Engineering College (A), Surampalem, Kakinada, Andhra Pradesh, India, Pincode: 533437. Email id : sunitha4949@gmail.com

[6]Assistant Professor, Department of Mathematics, Saveetha School of Engineering, Technical Sciences, Thandalam, Chennai, Tamilnadu, India, Pincode: 602105, Email id: lawanya.thangaraju@gmail.com

[7]Assistant Professor, Department of English, Koneru Lakshmaiah Education Foundation (KL Deemed to be University), Vaddeswaram, Guntur, Andhra Pradesh, India, Pin code: 522502. Email: dsurendra@kluniversity.in

Article History:

Received: *05-07-2024*

Revised: *20-08-2024*

Accepted: *02-09-2024*

Abstract:
Two-layered partial differential equations (PDEs) will be structured in this investigation using a variety of parallel mathematical techniques, such as Red Dull Gauss Seidel and Gauss Seidel. For the solution of this study, the two-layered PDEs of the elliptic and illustrated sorts were selected. Parallel Virtual Machine (PVM) is used on the correspondence among all chips of the Parallel Handling System. PVM is well-known for its item system, which enables several diverse computers to be used as flexible and easy-to-use concurrent handling resources. An evaluation of the parallel exhibition assessment including the Gauss Seidel and Red Gauss Seidel techniques in terms of execution time, speed, capability, ampleness, and transient execution will follow the graphical presentation of the mathematical results. Since the goal of this work was to create a successful Two-Dimensional PDE Solver (TDPDES), performance evaluations are simple. Numerous design and mathematical sectors will see an improvement in their examination and investigation systems thanks to this new, effective TDPDES technology..

Keywords: Partial differential equations in two dimensions; parallel numerical methods; hyperbolic, elliptic, and parabolic; performance evaluation.

1. Introduction

A significant role for partial differential equations is assumed in many fields of research and design. Due to the intricacy of settling partial differential equations, it is frequently wanted to utilize PCs to mathematically decide the arrangements. The limited contrast strategy, which repeats towards an answer given a few beginning conditions, is a common methodology for solving such equations. The

boring concept of the layout calculation makes this approach particularly well-suited to PC arrangement.

All things considered, large, high performance PC frameworks have been used to test limited contrast methodologies, and the arrangement techniques used on these frameworks are noteworthy. Nonetheless, as of late, the formation of enormously parallel PC systems has proposed an option computational methodology. The principal thought is to partition the game plan space into fragments, every one of which is doled out to an alternate processor in the parallel engineering. In this way, several small processors are used to settle a particular partial differential condition rather than a single large processor. Sometimes this parallel handling method can be more efficient than a single processor setup in terms of speed, cost, and ease of improvement.

Remorsefully, parallel dealing with structures haven't perceived their maximum capacity 100% of the time. In principle, a K-processor design will work roughly K times quicker than a solitary processor game plan, given that the issue can be sufficiently isolated. Wonderful execution utilizing parallel structures is impossible due to trouble deteriorating the issue and the correspondence practices that processors are by and large expected to finish.

This challenge includes addressing specific partial differential equations emerging in computational fluid components utilizing parallel PC systems. This presents the guess that partial differential equations can be addressed on parallel PC structures and that plans on these systems can be seriously settled with plans on huge, elite execution systems. For this audit, two sorts of partial differential equations were chosen: the two-layered Laplace's condition and the one-layered wave condition. The issue was spatially divided over the space of interest, and the restricted differentiation approach was applied. This is a regular issue rot technique that eliminates the "cost" of the parallel way to deal with taking care of interprocessor correspondence.

A strategy for dispensing issue sections to processors was created and executed for a grid associated parallel PC system. Participations between processors were distinguished and tried for the two-layered tabletop game life. Life's guidelines and the interprocessor practices for life act fairly in much the same way to confined separation answers for partial differential equations in that they raise the consistent qualities to the ensuing time step. Subsequently, the methods created to show the circle of life will make an interpretation of straightforwardly to the arrangement of partial differential equations. Utilizing the wave condition and Laplace's condition, that remnant is shown.

2. Literature Review

Mastoi et al. (2022) addresses the challenge of efficiently solving two-dimensional PDEs, which often arise in diverse fields such as fluid dynamics, heat transfer, and electromagnetism. The authors introduce the SM method as an innovative numerical technique for tackling these complex equations. The SM method stands out for its ability to offer accurate solutions while maintaining computational efficiency, which is crucial for handling large-scale problems encountered in practical applications. The essence of the SM method lies in its capacity to decompose the PDE problem into smaller subproblems, which are then solved iteratively. By breaking down the problem domain into manageable segments, the method facilitates the computation of solutions for complex PDEs with improved efficiency and accuracy. Mastoi present numerical results demonstrating the effectiveness

of the SM method in solving a variety of two-dimensional PDEs. Through comparative analysis with existing numerical techniques, the authors highlight the superior performance of the SM method in terms of solution accuracy and computational efficiency. Such empirical validation enhances the credibility of their proposed approach and underscores its potential for practical applications in diverse scientific and engineering disciplines.

Bähr (2021) researched effective time reconciliation strategies for straight allegorical partial differential equations, as recorded in his doctoral paper named "Productive time coordination techniques for direct illustrative partial differential equations with applications." Bähr's research focuses on the time integration aspect of solving PDEs, which complements the work in the realm of spatial discretization and solution techniques. addresses the crucial issue of selecting appropriate time integration schemes to ensure the stability, accuracy, and efficiency of numerical solutions for time-dependent PDEs. His research explores various time integration methods and evaluates their performance through theoretical analysis and numerical experiments. By devising efficient time integration strategies, Bähr contributes to the advancement of numerical techniques for solving parabolic PDEs, which are prevalent in fields such as heat conduction, diffusion processes, and reaction-diffusion systems.

Shang, Wang, and Sun (2022) propose a novel approach called the Deep Petrov-Galerkin method, as presented in their preprint titled "Deep Petrov-Galerkin method for solving partial differential equations." The Petrov-Galerkin method is a numerical technique for solving PDEs, and the authors introduce a deep learning framework to enhance its efficiency and accuracy. By integrating deep learning models into the Petrov-Galerkin method, they aim to achieve better solutions for a wide range of PDEs. This method potentially offers improvements in computational efficiency and accuracy compared to traditional numerical methods.

Grady et.al (2023) introduces another innovative approach in their paper "Model-parallel Fourier neural operators as learned surrogates for large-scale parametric PDEs," published in Computers & Geosciences. They propose the utilization of model-parallel Fourier brain administrators as scholarly substitutes for huge scope parametric PDEs. This method use brain organizations and Fourier investigation to effectively inexact the arrangements of complicated PDEs. By training neural networks to learn the underlying physics of the problem domain, they create surrogate models that can accurately predict the behaviour of the system under various parametric conditions. This approach demonstrates promise for addressing the computational challenges associated with solving large-scale PDEs in geosciences and related fields.

Guo, Cao, Liu, and Gao (2020) explore the integration of deep learning and physical constraints for solving PDEs in their paper "Solving partial differential equations using deep learning and physical constraints," published in Applied Sciences. Their approach combines deep learning architectures with physical principles to develop accurate and efficient solutions for PDEs. By incorporating knowledge of the underlying physics into the training process of neural networks, they aim to improve the generalization and stability of the models. This approach shows potential for addressing complex PDEs in various scientific and engineering applications.

3. SOLUTION METHODS

This round of life uses an exhibit made up primarily of zeros and a small number of ones, much like a checkerboard. They are referred to as "counters" There are eight neighbours for every counter. Four are symmetrically adjacent, while four are slantingly adjacent. John Conway, a mathematician at the College of Cambridge's Gonville and Caius School, provides the "hereditary standards" for life, which are as follows:

Consistencies. A counter that has two or three adjacent counters enables accommodations for future needs.

Deaths. Every counter that has four or more neighbours gets eliminated due to overcrowding. Every counter that has one neighbour or none at all faces disconnection.

births. A birth cell is any empty cell that is adjacent to exactly three neighbours, neither more nor less. In this cell, a counter is set at the next move.

It is important to recognise that all births and deaths happen simultaneously since all cells are evaluated at the same time. They aren't altered until the emphasis that follows.

Originally, the round of life was changed for a later PC. This successive programme gave rise to a parallel programme. The behaviour of this parallel programme is displayed below. Acknowledge that sixteen processors (numbered 0-15) are being utilized for usability. The fundamental communicator, Processor 0, is utilized to deteriorate every processor's part, as found in Figure 1.

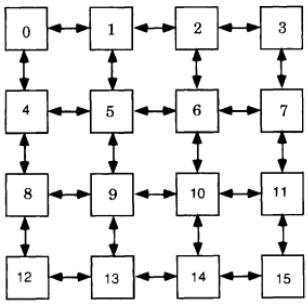

Figure 1: 16 processors arranged in a matrix evaluation layout

Processor 0 at first stores the whole one of a kind show; be that as it may, to lessen calculation time, it disseminates 1/16 of the main bunch for assessment among all processors. For every processor to have direct admittance to each of the eight of its neighbors, it gets an additional line of numbers encompassing the region outside its own sub-grid. Adjacent lines or sections from the close by processors make up these additional numbers. This should be visible in Figure 2..

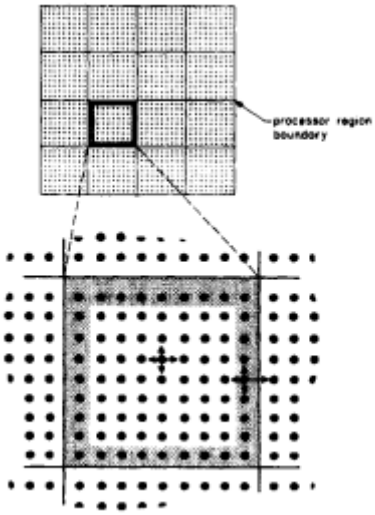

Figure 2: Matrix breakdown for each CPU, displaying shared rows and columns

The following is the interprocessor interaction algorithm employed in the previously discussed method.:

- Processor 0 holds the primary bunch.
- Processor 0 then sends every processor their piece alongside as far as possible lines.
- Every processor then copies their piece into a little concise bunch.
- The processors then check the neighbors of every cell in their piece bunches and record the fundamental changes in their temporary displays.
- Every processor then copies its fleeting display (which holds the progressions in general) into a last group.
- This last show just holds back the processor.

It's not actually the ideal approach to use a subsequent program's need for a parallel programme. Even though sixteen processors share the task of evaluating the lattice, processor 0 can still evaluate the correspondence. While other processors wait patiently for their assignments, processor 0 is left with an abundance of work.

This calls for an additional tactic. This approach suggests first characterising the problem before attempting to write a parallel programme that takes different processors into account. A second programme about the circle of life was written. This method determines each processor's four adjacent processors by looking at the processor format, such as a comparable sixteen. For the purpose of updating their own clusters, these neighbours would need the exterior lines and segments of the processor.

For example, processor S's neighbors in the handling configuration displayed in Figure 1 are north = 1, south = 9, east = 6, and west = 4. Processors 4 and 6 would get the farthest left and right fragments independently, while Processor 1 would acknowledge S's most essential line and Processor 9 would acknowledge S's last section. Since the processors can rapidly accept their messages and complete their undertakings, this altogether lessens the costs related with correspondence. Coming up next is the computation for this metho.:

- Processor 0 is accountable for the primary bunch.
- Processor 0 sends every processor their piece alongside as far as possible lines.
- Every processor duplicates its piece into a little fleeting bunch.
- The processors then check the neighbors of every cell in their piece groups and record any huge changes in their temporary shows.
- Every processor duplicates its temporary group, which contains the progressions as a whole, into a last display.
- The last show just incorporates the proposition and not
- handling 0 reassembles and prints the monstrous show in the wake of getting the last evaluated group from each handling;
- beginning with this cycle, the processors recognize their neighbors.
- To fulfill the breaking point conditions, every processor sends the particular lines or sections to its neighbors.
- The processors assess their segments as before; processor 0 duplicates every processor's part, returns the monstrous group together, and prints.
- the emphasess go on with the processors conveying among themselves

4. PARTIAL DIFFERENTIAL EQUATIONS

The wave condition in one dimension is provided by:

$$\frac{1}{c^2} * \frac{\partial \Psi}{\partial t^2} - \frac{\partial \Psi}{\partial x^2} = 0 \qquad (1)$$

where $c_{.p} = c_{.p}(x,t)$, x = position, t = time

The limited distinction condition provides one estimate of this partial differential condition.:

$$\Psi_i(t + \Delta t) = 2\Psi_i(t) - \Psi_i(t - \Delta t) + \tau^2[\Psi_{i-1}(t) + 2\Psi_i(t) + \Psi_{i+1}(t)] \qquad (2)$$

Where $\tau = \frac{c\Delta t}{\Delta x}$

Take note of the fact that this low contrast condition in particular resembles a collection of inherited norms similar to those used in the round of life. Given the one-dimensional nature of the scenario, the arrangement is stored as a section of numbers rather than as a two-dimensional show. Each part of the segment deals with the value of c.p at some point in time, let's say t. The two initial phases of the evaluation cycle are distinguished by the introductory conditions provided. (The characteristics for c.p at time 0 and time O+Llt are provided in these two opening segments.) The main segment is used as a final emphasis correlation, and the limited contrast condition is applied in the following portion. These two programme segments collaborate to determine the values for c.p at time 2Llr. A third section assigns scores to these attributes. The programme repeats the exchange after concluding the process of determining the requirements for the third segment.

You can examine the limited distinction condition at the relevant lines and sections.:

$$\Psi[iIn + 1] = 2\Psi[iIn - 1] + \tau^2(\Psi[i + 1In] - 2\Psi[iIn] + \Psi[i - 1In]) \qquad (3)$$

In essence, the programme examines the characteristics of the section's neighbouring cells as well as the emphasis from the past and present. Two distinct approaches were used to provide the wave

condition organisation methodology. Finding a parallel calculation using a sequential basis was the first step; coming up with a rigorously parallel calculation was the second.

- The parallel computation for the wave condition, based on a consecutive basis, is as follows:
- Processor 0 contains two distinct portions.
- Processor 0 supplies the basic limit numbers and its share of the next section to each processor.
- Processor 0 gets every processor's piece, reassembles, and prints after that.
- Every processor duplicates its portion into a temporary cluster.
- Every processor checks out at its portion and rolls out the essential enhancements in this temporary cluster.
- emphasise the next step.

Here is an algorithm that runs in strict parallel.:

- Processor 0 is responsible for maintaining two distinct sections.
- It additionally supplies every processor with its part of the following area and as far as possible numbers.
- Every processor copies its part into a brief bunch.
- Every processor looks at its part and carries out the huge upgrades in this transient show.
- Processor 0 gets a copy of every processor's part, reassembles it, and prints.
- Every processor distinguishes its neighbor beginning with this cycle.
- Every processor sends its first and last numbers to its adjacent neighbors, separately.
- Every processor assesses its part as it did already.
- Processor 0 gets a copy again to print.

The Laplace's condition in two dimensions is given by:

$$\nabla^2 \emptyset = \frac{\partial^2 \emptyset}{\partial x^2} + \frac{\partial^2 \emptyset}{\partial y^2} = 0 \qquad (4)$$

Where $\emptyset = \emptyset(x, y)$

For the first Jacobi cycle of the limited distinction condition, the iterative approach was used. For the next emphasis, the Jacobi update approach is composed as:

$$\emptyset_i^{(k)} = \frac{1}{4}\left[\emptyset_{(i-k)}^{(k-1} + \emptyset_{(i+x)}^{(k-1)} + \emptyset_{(i+y)}^{(k-1)}\right] \qquad (5)$$

where time is addressed by the superscript and position by the addendum. Laplace's problem has a two-dimensional solution, which is stored as a lattice. According to the specified limit criteria, a distinct lattice is filled with numbers before any evaluation rakes are placed. In its latter opportunity focus, each cell's update is essentially the normal of its four symmetrical neighbours. Initially, in order to track the correspondences among neighbouring cells, the round of life was carried out..

In essence, the restricted contrast criterion for Laplace's scenario is just another configuration of inherited norms for the circle of life. The programme just needs to take the average of four neighbours for the next cycle, as opposed to examining eight neighbours. The round of life showed how the

processors' correspondence techniques may be applied to solve two-dimensional partial differential equations.

5. PARALLEL COMPUTING

Three classifications can be used to group the parallel PC design: Flynn's scientific classification, Quinn arrangement, and Cheong characterization. Since the PVM structure sticks to the common memory, cream, and message passing necessities, applications can pick the best figuring model for either the whole program or certain sub-computations. Scalar machines, conveyed and shared-memory multiprocessors, vector supercomputers, and engines with unique reason plans were among the parts taken care of, empowering the utilization of the best figuring resource for each fragment of an application. A few large and intricate applications, such as global ecological demonstration, liquid element reenactments, and climate expectation applications, depend on this adaptability.

The PVM framework operates on an equipment base made up of different machine designs, such as vector machines, multiprocessors, and single computer processor frameworks. At least one network, which may differ from one another, connects this processing component. For example, a PVM execution may operate over an Ethernet, Web, or fibre optic network.

PVM codes can be written in the dialects of C, C++, and FORTRAN. UNIX dialects are used as a working framework to do this assignment. Commonly, a client begins one copy on one errand from a PC in the host pool to execute an application. In PVM, task-to-task correspondence is finished through message passing. Message passing is an assortment of undertakings that figure utilizing their own neighborhood memory. Different tasks can run on comparable real computers or over a variable number of machines. Businesses exchange information by sending and receiving messages via correspondence. Information flow often necessitates each encounter to complete beneficial objectives. For example, a send action ought to correspond with a corresponding get action.

6. PRACTICAL COMPUTING TO RESOLVE PDE NUMERICAL METHODS

We encountered problems with massive specialised concerns, such as lengthy cycle calculation times, computationally challenging code, or extensive data collecting in the usage arrangement for which we are writing the code. Numerous problems force the cycle to accelerate and reduce performance. The best way to handle this problem is to use parallel processing. A graphics processing unit (GPU) is used for widely applicable parallel handling applications in universally usable computation on GPUs (GPGPU). The primary safeguards against parallel processing are the absence of requirements or correspondences between assignments. One of the main concerns with GPU code is vectorization. The majority of GPU operations operate in a vectorized manner, allowing an activity to be executed on up to four qualities simultaneously. Given that almost all basic information is a vector (either two-, three-, or four-dimensional), the value of generating yield from two distinct free information sources is significant in scenarios. Data will be transported from MATLAB's work area to GPU gadget memory at the moment when MATLAB's GPU-enabled capabilities are implemented on the GPU. A specific cluster type is assigned by the order "gpuArray" to such information, at which time GPU-enabled capabilities can operate on it. The "gather" order returns the determined results to MATLAB's work area once they have been stored in the GPU. Figure illustrates the MATLAB GPU processing approach.

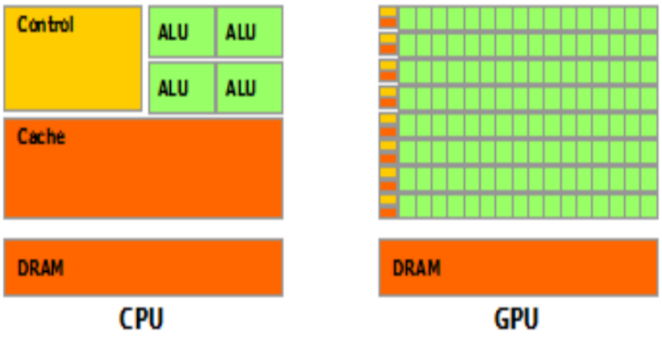

Figure 3: Procedures used in GPU computing

One advantage of building computer programmes in MATLAB GPU is that users can easily use GPU registering by adding very few additional commands to their custom MATLAB scripts. The fact that the majority of MATLAB's limited capabilities are GPU-powered and that MATLAB's GPU-based computation is not exactly the same as that of CUDA-written routines are among the obstacles.

7. PARALLEL PERFORMANCE EVALUATION

The parallel calculation's performance will be broken down in terms of execution time, speed, productivity, viability, and real-world performance. These are the characteristics of the estimations.:

$$Speedup: s(p) = \frac{t_1}{t_p}$$

$$Efficiency: E(P) = \frac{S(P)}{Pt_p}$$

$$Effectivenss : F(p) = \frac{s(p)}{pt_r} = \frac{E_p}{t_p}$$

$$Temporal\ performance : L(p) = t_p^{-1} = \frac{1}{t_p}$$

t_1= execution time for a single processor and

t_p = execution time using p parallel processors.

that as the quantity of processors expands, the execution time diminishes. The decrease in execution time with an expansion in processor count is additionally seen while tackling explanatory and exaggerated issues. shows how the speedup increments with the quantity of processors added. The contention for this is that the common memory order makes it less burdening to enter a group of workstations. A parallel program's productivity is an element of processor use. that as the quantity of processors increments, efficiency diminishes. Capability is, as is notable, the proportion of speedup to processor count. Hence, capability is a performance that is firmly connected to speed increase. As the quantity of processors expands, the feasibility increments. The reasonability condition relies upon the speed increase; as this speed increase builds, so will the ampleness.

that the connection between the world performance outline and the quantity of processors increments. This is on the grounds that the quantity of processors is expanding as the execution time diminishes. It is generally accepted that rising the quantity of processors works on the performance of parallel estimations as far as execution time, speed, efficiency, adequacy, and transient performance. The performance of parallel figuring is continually affected by correspondence and execution times. The robust Red Dark Gauss Seidel is thought to be suitable for parallel execution on PVM, where information disintegration is carried out concurrently at every time step. Applications of the PVM framework include the replication of atomic elements, the review of superconductivity, calculations of distributed fractals, network computations, and the demonstration of simultaneous figures in study halls..

8. RESULTS

Using a 64 center nCUBE 2 PC structure and a parallel C program, the answer for the one-layered wave condition was found. The time expected for the program to work on different quantities of processors was contrasted with the time expected while utilizing only one processor. Table 1 contains these subtleties.

Table 1: The sequential algorithm Created in Parallel

Number of Processors	Total Time	Speedup
1	25.12485	
2	12.585786	1.886
4	5.932076	3.982
8	2.104460	7.945
16	1.600458	16.408
32	.788542	28.685
64	.428035	54.295

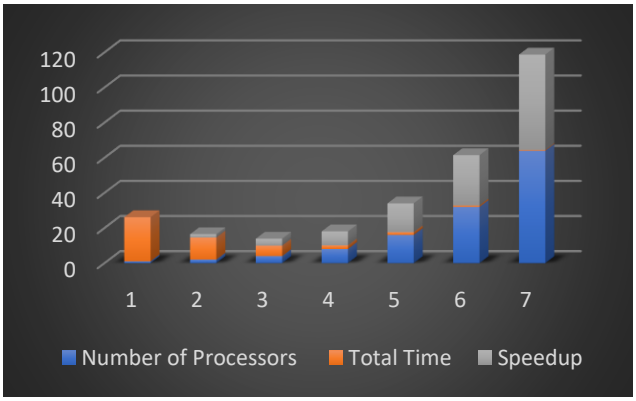

Figure 4: Sequential Algorithm Made Parallel

A framework with N processors should ideally result in a programme speedup of N. Correspondence costs will eventually reduce the speedup. The speed improvements (such 55.294 for 64 CPUs) were really encouraging.

Now note that the correlation shown above was measured against the amount of time it took a single processor to perform a parallel computing programme.

The parallel times should be compared against a strictly consecutive programme in order to determine any speedup. For the one-dimensional wave condition, a sequential programme was run, and the run time was noted. Table 2 shows how much faster the parallel rendition was than the consecutive adaption. In this case, the speedups were remarkably faster than in the past..

Table 2: Acceleration of the Parallel Process

Number of Processors	Speedup
2	2.185
4	4.555
8	8.585
16	16.849
32	32.455
64	60.465

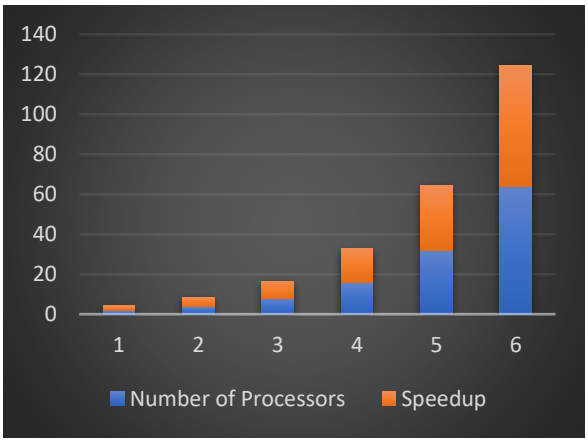

Figure 5 : Speedup of Parallel Algorithm

Tests are presently being attempted to test the responsiveness of our answer for the cluster size. Instinctively, our answer ought to be more productive the bigger the exhibit. Primer outcomes support this, however more tests are arranged.

9. CONCLUSIONS

The C programming language was utilized in parallel to do the round of life on a 64-center point nCUBE 2 PC system. The one-layered wave condition and the two-layered Laplace's condition were then replied pair utilizing this execution with not very many adjustments. The initial results are very encouraging, despite the fact that performance data are yet incomplete. The wave condition arrangement speedups suggest that correspondence prices won't be a significant barrier to numerically solving many partial differential equations on parallel PC frameworks. Further study will involve trying the executions under a wide range of limit conditions and gathering performance data. Findings on the solution to Laplace's scenario will soon be available.

References

[1] Götschel, S., & Weiser, M. (2019). Compression challenges in large scale partial differential equation solvers. *Algorithms*, *12*(9), 197.
[2] Bashkov, E. A., Dmitrieva, O. A., Huskova, N. H., Vishnevskyi, S. Y., Kotyra, A., & Ormanbekova, A. (2022, December). Parallel implementation of evolutionary partial differential equations by collocation optical-electronic schemes. In *Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments 2022* (Vol. 12476, pp. 236-241). SPIE.
[3] Nielsen, A. S., Brunner, G., & Hesthaven, J. S. (2018). Communication-aware adaptive Parareal with application to a nonlinear
[4] equations. *Journal of Computational Physics*, *371*, 483-505.
[5] Heene, M. (2018). A massively parallel combination technique for the solution of high-dimensional PDEs.
[6] Ni, N., & Dong, S. (2023). Numerical computation of partial differential equations by hidden-layer concatenated extreme learning machine. *Journal of Scientific Computing*, *95*(2), 35.
[7] Aditya, K., Gysi, T., Kwasniewski, G., Hoefler, T., Donzis, D. A., & Chen, J. H. (2019). A scalable weakly-synchronous algorithm for solving partial differential equations. *arXiv preprint arXiv:1911.05769*.
[8] Lederman, C., Martin, R., & Cambier, J. L. (2018). Time-parallel solutions to differential equations via functional optimization. *Computational and Applied Mathematics*, *37*, 27-51.
[9] Magee, D. J., Walker, A. S., & Niemeyer, K. E. (2021). Applying the swept rule for solving explicit partial differential equations on heterogeneous computing systems. *The Journal of Supercomputing*, *77*, 1976-1997.
[10] Asgari, B., Hadidi, R., Krishna, T., Kim, H., & Yalamanchili, S. (2021). Efficiently solving partial differential equations in a partially reconfigurable specialized hardware. *IEEE Transactions on Computers*, *70*(4), 524-538.
[11] Magoulès, F., & Gbikpi-Benissan, G. (2018). Asynchronous parareal time discretization for partial differential equations. *SIAM Journal on Scientific Computing*, *40*(6), C704-C725.
[12] Chen, T., Botimer, J., Chou, T., & Zhang, Z. (2020). A 1.87-mm 2 56.9-GOPS accelerator for solving partial differential equations. *IEEE Journal of Solid-State Circuits*, *55*(6), 1709-1718.
[13] Carlson, M., Kirby, R. M., & Sundar, H. (2020, June). A scalable framework for solving fractional diffusion equations. In *Proceedings of the 34th ACM International Conference on Supercomputing* (pp. 1-11).
[14] Fabiani, G., Calabrò, F., Russo, L., & Siettos, C. (2021). Numerical solution and bifurcation analysis of nonlinear partial differential equations with extreme learning machines. *Journal of Scientific Computing*, *89*, 1-35.
[15] Gloster, A. (2021). GPU methodologies for numerical partial differential equations. *arXiv preprint arXiv:2101.06550*.
[16] McDonald, E., Pestana, J., & Wathen, A. (2018). Preconditioning and iterative solution of all-at-once systems for evolutionary partial differential equations. *SIAM Journal on Scientific Computing*, *40*(2), A1012-A1033.
[17] Mastoi, S., Ganie, A. H., Saeed, A. M., Ali, U., Rajput, U. A., & Mior Othman, W. A. (2022). Numerical solution for two-dimensional partial differential equations using SM's method. *Open Physics*, *20*(1), 142-154.
[18] Bähr, M. (2021). *Efficient time integration methods for linear parabolic partial differential equations with applications* (Doctoral dissertation, BTU Cottbus-Senftenberg).
[19] Shang, Y., Wang, F., & Sun, J. (2022). Deep Petrov-Galerkin method for solving partial differential equations. *arXiv preprint arXiv:2201.12995*.
[20] Grady, T. J., Khan, R., Louboutin, M., Yin, Z., Witte, P. A., Chandra, R., ... & Herrmann, F. J. (2023). Model-parallel Fourier neural operators as learned surrogates for large-scale parametric PDEs. *Computers & Geosciences*, 105402.
[21] Guo, Y., Cao, X., Liu, B., & Gao, M. (2020). Solving partial differential equations using deep learning and physical constraints. *Applied Sciences*, *10*(17), 5917.