Procedia Computer Science

Volume 80, 2016, Pages 1917–1926

ICCS 2016. The International Conference on Computational Science

A Robust Technique to Make a 2D Advection Solver Tolerant to Soft Faults

Peter Strazdins[1], Brendan Harding[2], Chung Lee[1], Jackson R. Mayo[3], Jaideep Ray[3], and Robert C. Armstrong[3]

[1] Research School of Computer Science, Australian National University, Canberra, ACT, Australia
peter.strazdins@cs.anu.edu.au, brian.lee@anu.edu.au

[2] Mathematical Sciences Institute, Australian National University, Canberra, ACT, Australia
brendan.harding@anu.edu.au

[3] Sandia National Laboratories, Livermore, California, USA
jmayo@sandia.gov, jairay@sandia.gov, rob@sandia.gov

Abstract

We present a general technique to solve Partial Differential Equations, called robust stencils, which make them tolerant to soft faults, i.e. bit flips arising in memory or CPU calculations. We show how it can be applied to a two-dimensional Lax-Wendroff solver. The resulting 2D robust stencils are derived using an orthogonal application of their 1D counterparts. Combinations of 3 to 5 base stencils can then be created. We describe how these are then implemented in a parallel advection solver. Various robust stencil combinations are explored, representing tradeoff between performance and robustness. The results indicate that the 3-stencil robust combinations are slightly faster on large parallel workloads than Triple Modular Redundancy (TMR). They also have one third of the memory footprint. We expect the improvement to be significant if suitable optimizations are performed. Because faults are avoided each time new points are computed, the proposed stencils are also comparably robust to faults as TMR for a large range of error rates. The technique can be generalized to 3D (or higher dimensions) with similar benefits.

Keywords: exascale computing, fault-tolerance, partial differential equations, robust stencils, advection equation, parallel computing, resilient computing

1 Introduction

There is an increasing need for the ability to be resilient to faults for various scenarios of computations [10, 3]. Two kind of faults may occur: hard faults, arising from the transient or permanent failure of a software or hardware resource, and soft or silent faults, where the computation continues but errors are introduced into data. The detection and recovery for

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2016

doi:10.1016/j.procs.2016.05.505

either generally requires different approaches, with soft faults being generally regarded as being more difficult [5].

In most situations, soft faults can lead to erroneous results; in some cases, this may not be obvious. Soft faults normally manifest as a (random) bit flip in a data item: depending on the importance of the item, and the significance of the bit, the fault maybe anything from unnoticeable, insignificant, significant but not obvious, obvious or catastrophic. These can arise either when the item is stored at some location in the memory hierarchy, while the item is on a datapath, or while a new version of the item is being created in the CPU.

Soft errors are of concern when the probability of error reaches a point where the overall chances of completing the computation correctly becomes uncertain. Various factors affect this: the length of the computation, the size of the system (exascale computing), and error rates in each component. The latter can be exacerbated by adverse operating conditions, such as adverse operating environments, low-quality components, low power. The last two are potentially important in the context of exascale computing, as low cost components can keep the purchase cost down, and minimizing operational power of components lowers the running costs.

Checkpointing can be used to mitigate soft faults, but this normally requires duplication for detection, and triplication for recovery. It also requires a *resilient store* which, particularly if distributed, is difficult to implement and expensive to access [4].

Algorithm-based fault tolerance (ABFT) can also be employed to mitigate soft faults. The baseline is Triple Modular Redundancy (TMR) [6], which involves evolving three versions of the computation (and data), with periodic comparison of the corresponding data items. A voting scheme is normally used to determine the correct value, which must be replicated. The normal overhead, compared to the original computation, is at least a factor of three, both in time and space. However, the method is relatively simple and general, and is extremely robust: provided the error does not occur in the voting phase itself, a corruption is required on at least two of the three versions of the same data item.

However, application-specific versions of ABFT have the potential to provide the same benefit for lower cost to TMR. Specifically, this is because the level of redundancy can be less than 3. For example, in fault-tolerant versions of the Sparse Grid Combination Technique, the redundancy can be as little as 20% [2]. In this case, some (small) loss of accuracy must be tolerated.

In this paper, we extend the concept of *robust stencils*, introduced in [9], a form of ABFT which is exact, but has the potential of reduced overheads to TMR. It can be applied in principle to any explicit solution of a Partial Differential Equation (PDE). It can be implemented with neglible memory overhead. While its computational overhead must be at least 3 in terms of overall FLOPS, as most PDE solvers are memory-bound, its overhead in terms of memory accesses can be reduced to a factor of 5/3, in the case of simple stencils. One such stencil arises from the 2D advection equation via the Lax-Wendroff method [8], which is the object of study in this paper.

To the best of our knowledge, no other work addresses the issue of making 2D (or higher) PDE solvers tolerant to soft faults, in either a similar, or in a different and superior (in all aspects), fashion.

This paper is organized as follows. Section 2 gives the derivation of the 2D robust stencils from their 1D counterparts. A description of how the 2D stencils were implemented is given in Section 3. The performance of the stencils, compared with the baseline (TMR), is given in Section 4. This section details the error and speed in the fault-free case, and the robustness in the fault-injected case. Conclusions and future work are summarized in Section 5.

2 Derivation of stencils

The one dimensional Lax-Wendroff stencil [8] for solving the advection equation $\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0$ for fixed $a \in \mathbb{R}$ and $u(x,t) : \Omega \subset \mathbb{R} \times [0,\infty) \to \mathbb{R}$ with appropriate initial and boundary conditions is

$$u(x,t+\Delta t) \approx u(x,t) + \frac{a\Delta t}{2\Delta x}(u(x-\Delta x,t) - u(x+\Delta x,t)) + \frac{a^2\Delta t^2}{2\Delta x^2}(u(x-\Delta x,t) - 2u(x,t) + u(x+\Delta x,t))\,.$$

It is well known this scheme is 2nd order consistent and stable for $\frac{a\Delta t}{\Delta x} \leq 1$. In a finite difference implementation with $u_i^n := u(i\Delta x, n\Delta t)$ and $U := \frac{a\Delta t}{\Delta x}$ we may write this as

$$u_i^{n+1} \approx u_i^n + \frac{U}{2}(u_{i-1}^n - u_{i+1}^n) + \frac{U^2}{2}(u_{i-1}^n - 2u_i^n + u_{i+1}^n)\,.$$

We define the shift operator S_k for which $S_k u(x,t) = u(x+k\Delta x,t)$, and thus $S_k u_i^n = u_{i+k}^n$. With this we define the 'normal' Law-Wendroff stencil operator by

$$\begin{aligned} S_N &:= S_0 + \frac{U}{2}(S_{-1} - S_1) + \frac{U^2}{2}(S_{-1} - 2S_0 + S_1) \\ &= \left(\frac{U}{2} + \frac{U^2}{2}\right) S_{-1} + (1-U^2)S_0 + \left(-\frac{U}{2} + \frac{U^2}{2}\right) S_1\,, \end{aligned}$$

and thus we may write $u_i^{n+1} \approx S_N u_i^n$. A second stencil may be formed by simply replacing Δx with $2\Delta x$. We refer to this as the 'wide' stencil, in particular we define

$$\begin{aligned} S_W &:= S_0 + \frac{U}{4}(S_{-2} - S_2) + \frac{U^2}{8}(S_{-2} - 2S_0 + S_2) \\ &= \left(\frac{U}{4} + \frac{U^2}{8}\right) S_{-2} + \left(1 - \frac{U^2}{4}\right) S_0 + \left(-\frac{U}{4} + \frac{U^2}{8}\right) S_2\,. \end{aligned}$$

As this is effectively the Lax-Wendroff stencil on a coarser grid the consistency and stability follows immediately. We define a third stencil we refer to as the 'far' stencil which is generated by the operator

$$\begin{aligned} S_F &:= \frac{-S_{-3} + 9S_{-1} + 9S_1 - S_3}{16} + \frac{U}{6}(S_{-3} - S_3) + \frac{U^2}{16}(S_{-3} - S_{-1} - S_1 + S_1) \\ &= \left(\frac{U}{6} + \frac{U^2-1}{16}\right) S_{-3} + \frac{9-U^2}{16} S_{-1} + \frac{9-U^2}{16} S_1 + \left(-\frac{U}{6} + \frac{U^2-1}{16}\right) S_3\,. \end{aligned}$$

The second order consistency of this stencil is easily shown via a Taylor series expansion (with appropriate smoothness of u). Further, one may show the S_F stencil is stable for $a\Delta t \leq 3\Delta x$ with a standard von Neumann stability analysis. We leave the proofs of stability of the stencils for future publication [1].

The fundamental idea of robust stencils is to compute many of these different stencils, each using a different subset of neighbouring function values, so that given an isolated point whose value is affected by a silent fault then at least one of the three stencils will provide the correct

[1] Interested readers should contact author Mayo for the proofs.

result. Of course the difficulty is in having a robust way of determining which is correct at runtime.

For the one dimensional problem these three stencils are not sufficient as the first two share the centre point. Note that this can be avoided by an outlier detection techniqe [9]. However we may use these 3 stencils to derive several different stencils for the advection equation in higher dimensions to develop robust algorithms.

Fix $d \in \mathbb{N}$ and consider d-dimensional advection equation $\frac{\partial u}{\partial t} + \mathbf{a} \cdot \nabla u = 0$, where $\nabla u = \left(\frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_d}\right)$, for fixed $a \in \mathbb{R}^d$ and $u(\mathbf{x}, t) : \Omega \subset \mathbb{R}^d \times [0, \infty) \to \mathbb{R}$ with appropriate initial and boundary conditions. To solve the d-dimensional advection equation one may use tensor products of the one dimensional stencils above. For example, given $\alpha \in \{N, W, F\}^d$ then we define the stencil

$$S_\alpha := \bigotimes_{k=1}^{d} S_{\alpha_k} .$$

For brevity we will often write $S_\alpha = S_{(\alpha 1,\ldots,\alpha_d)}$ as just $S_{\alpha_1 \cdots \alpha_d}$. For example, with $d = 2$ and $\alpha = (N, N)$ then $S_\alpha = S_N \otimes S_N$ which we denote with S_{NN}. Further, with $U_i = \frac{a_i \Delta t}{\Delta x_i}$, S_{NN} may be expanded as

$$\begin{aligned} S_{NN} = &\left(\frac{U_1}{2} + \frac{U_1^2}{2}\right)\left(\left(\frac{U_2}{2} + \frac{U_2^2}{2}\right) S_{-1} \otimes S_{-1} + (1 - U_2^2) S_{-1} \otimes S_0 + \left(-\frac{U_2}{2} + \frac{U_2^2}{2}\right) S_{-1} \otimes S_1\right) \\ &+ (1 - U_1^2)\left(\left(\frac{U_2}{2} + \frac{U_2^2}{2}\right) S_0 \otimes S_{-1} + (1 - U_2^2) S_0 \otimes S_0 + \left(-\frac{U_2}{2} + \frac{U_2^2}{2}\right) S_0 \otimes S_1\right) \\ &+ \left(-\frac{U_1}{2} + \frac{U_1^2}{2}\right)\left(\left(\frac{U_2}{2} + \frac{U_2^2}{2}\right) S_1 \otimes S_{-1} + (1 - U_2^2) S_1 \otimes S_0 + \left(-\frac{U_2}{2} + \frac{U_2^2}{2}\right) S_1 \otimes S_1\right) , \end{aligned}$$

which, with $u^n_{i,j} := u((i\Delta x_1, j\Delta x_2), j\Delta t)$, leads to the update formula

$$\begin{aligned} u^{n+1}_{i,j} = &\left(\frac{U_1}{2} + \frac{U_1^2}{2}\right)\left(\left(\frac{U_2}{2} + \frac{U_2^2}{2}\right) u^n_{i-1,j-1} + (1 - U_2^2) u^n_{i-1,j} + \left(-\frac{U_2}{2} + \frac{U_2^2}{2}\right) u^n_{i-1,j+1}\right) \\ &+ (1 - U_1^2)\left(\left(\frac{U_2}{2} + \frac{U_2^2}{2}\right) u^n_{i,j-1} + (1 - U_2^2) u^n_{i,j} + \left(-\frac{U_2}{2} + \frac{U_2^2}{2}\right) u^n_{i,j+1}\right) \\ &+ \left(-\frac{U_1}{2} + \frac{U_1^2}{2}\right)\left(\left(\frac{U_2}{2} + \frac{U_2^2}{2}\right) u^n_{i+1,j-1} + (1 - U_2^2) u^n_{i+1,j} + \left(-\frac{U_2}{2} + \frac{U_2^2}{2}\right) u^n_{i+1,j+1}\right) . \end{aligned}$$

Similarly the S_{WN} stencil leads to the update formula

$$\begin{aligned} u^{n+1}_{i,j} = &\left(\frac{U_1}{4} + \frac{U_1^2}{8}\right)\left(\left(\frac{U_2}{2} + \frac{U_2^2}{2}\right) u^n_{i-2,j-1} + (1 - U_2^2) u^n_{i-2,j} + \left(-\frac{U_2}{2} + \frac{U_2^2}{2}\right) u^n_{i-2,j+1}\right) \\ &+ \left(1 - \frac{U_1^2}{4}\right)\left(\left(\frac{U_2}{2} + \frac{U_2^2}{2}\right) u^n_{i,j-1} + (1 - U_2^2) u^n_{i,j} + \left(-\frac{U_2}{2} + \frac{U_2^2}{2}\right) u^n_{i,j+1}\right) \\ &+ \left(-\frac{U_1}{4} + \frac{U_1^2}{8}\right)\left(\left(\frac{U_2}{2} + \frac{U_2^2}{2}\right) u^n_{i+2,j-1} + (1 - U_2^2) u^n_{i+2,j} + \left(-\frac{U_2}{2} + \frac{U_2^2}{2}\right) u^n_{i+2,j+1}\right) . \end{aligned}$$

The remaining 7 stencils in two dimensions, namely S_{FN}, S_{NW}, S_{WW}, S_{FW}, S_{NF}, S_{WF} and S_{FF}, are also similarly obtained. In higher dimensions the total number of stencils obtained in this way is clearly 3^d. Such stencils can be viewed as an application of operator splitting techniques (see for example the relevant chapter in [7]). In this case the splitting approximation

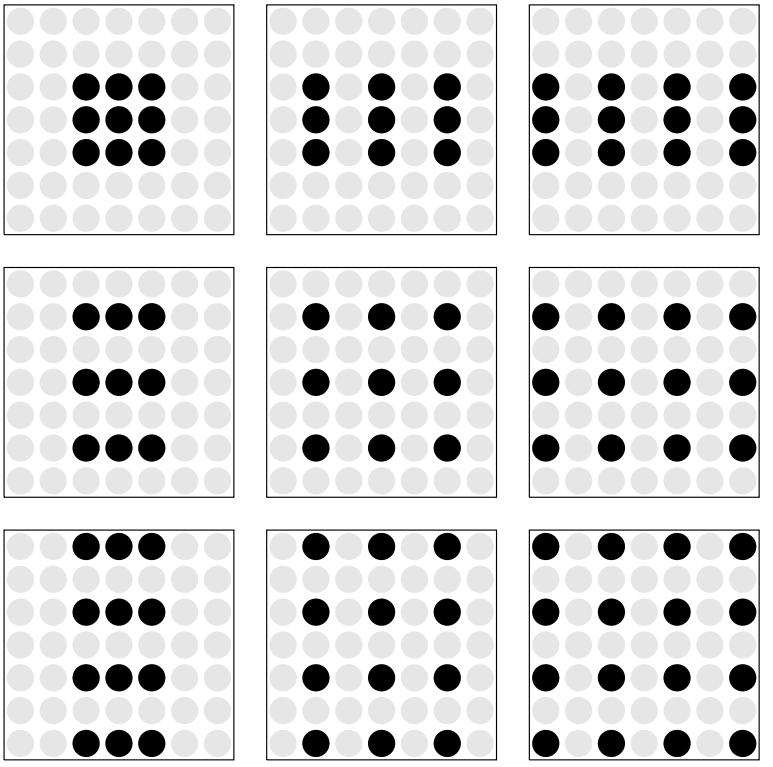

Figure 1: A depiction of the nodes used in each stencil with the centre being $u^n_{i,j}$. From left to right, top to bottom the stencils depicted are NN, WN, FN, NW, WW, FW, NF, WF and FF.

is exact because the differential operators $a_k \frac{\partial}{\partial x_k}$ commute with each other. It is also immediate that each of these stencils are second order consistent and stable when $U_k \leq 1$ for all $k = 1, \ldots, d$ (that is $\Delta t \leq \min_k a_k \Delta x_k$). For stencils with $\alpha \in \{W, F\}^d$ this could be relaxed but we would generally like to choose Δt such that all of the stencils are stable.

A robust approach to finite difference computations in the presence of silent errors (e.g. bit flips) involves computing several of the above stencils described above and taking the median. The points used by each of the stencils when applied to $u^n_{i,j}$ are depicted in Figure 1. With these it is easy to verify how many times each neighbour is used in each collection of stencils. For example, suppose we compute 5 stencils for which no more than 2 use any one of the neighbouring function values (including the centre of the stencil), then given a sufficiently large error one of the remaining 3 will be selected as the median. As a result one can effectively avoid any errors that affect isolated floating point numbers stored in memory. As an example, five such stencils in two dimensions would be those derived from the S_{NN}, S_{WW}, S_{WF}, S_{FW} and S_{FF} stencils. An example using only three stencils is S_{WW}, S_{WF} and S_{FW}, or alternatively S_{FF}, S_{WF} and S_{FW}. Here no one neighbour is in more than one stencil. Another example using seven stencils is S_{NW}, S_{NF} S_{WN}, S_{WW}, S_{WF}, S_{FN} and S_{FW}. Here no one neighbour is used in more than 3 of the seven stencils. Figure 2 depicts some of the examples mentioned here.

Whilst the derivation of robust stencils above applies to dimensions $d \geq 2$ the discussion in the remainder of the paper will be restricted to the $d = 2$ case.

S_{**}	N	W	F
N			
W		*	*
F		*	

S_{**}	N	W	F
N	*		
W		*	*
F		*	*

S_{**}	N	W	F
N		*	*
W	*	*	*
F	*	*	

Figure 2: A depiction of some of the robust stencil combinations for 2 dimensional advection selected from the nine stencils available. From left to right we have examples of robust stencil combinations consisting of three, five and seven stencils respectively.

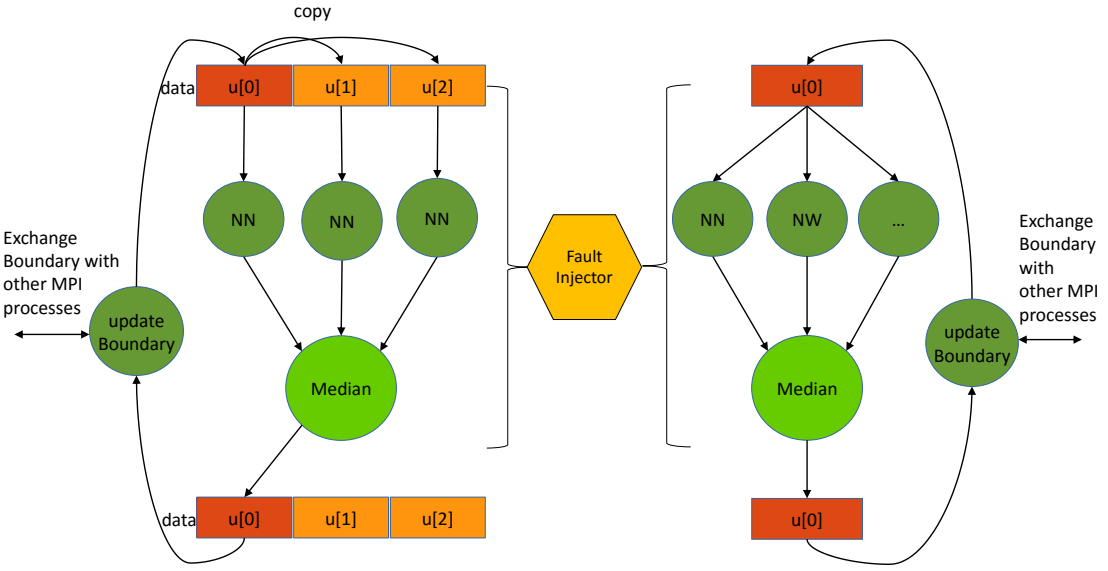

Figure 3: Implementation of both TMR and Robust Stencils

3 Implementation

We have implemented the stencils, as expressed in the previous section, coding them in a straightforward way in C++. Due to time constraints, we have not optimized these codes. It should be noted that, due to their complexity, robust stencils have a greater potential for optimization than simple stencils (such as NN).

For example, with stencil combinations, each element gets reloaded for each stencil. This can be avoided if we load the currently being processed elements into a 5×3 local array (or better still, block of registers): each stencil can access those elements without any extra movement further down the memory hierarchy. This will mean that the amount of memory accesses of any combination would be the same as a single (NN) stencil. As (single) stencil computations are memory-bound, the overhead of robust stencils can now be made very low.

Figure 3 demonstrates our implementation of TMR and robust stencils for the 2D advection solver.

Our implementation of TMR only uses the NN stencil, that provides the best accuracy, three times. As depicted in Figure 3, at the beginning, the initial/updated field data is duplicated, and, after computation, and the appropriate version of each element is stored in the first array.

This is then copied to the other arrays.

For both TMR and robust stencils, we take the median when selecting multiple results, with Inf and NaN values filtered beforehand. It should be noted that the result selection phase, in either case, while critical for fault tolerance, affects performance very little.

Our 2D advection solver [1, 11] offers full MPI parallelization on a 2D logical process grid, i.e. a $p \times q$ grid where $p, q > 0$. TMR is given the same process grid as the robust stencils: this means that each process will have three local arrays, and processes them serially. This means that the same compute resources are given to both, and TMR requires three times the memory at each resource.

The alternate possibility is to give TMR a $3 \times p \times q$ grid, where three times the compute resources is given to TMR but the memory requirement per compute resources remains constant. While desirable for hard faults, this is undesirable for soft faults as the voting stage becomes highly communication intensive.

Under the selected scheme for TMR, the communication volume for the two methods is equivalent. Robust stencils for 2D advection requires a halo of width 3, and TMR requires the exchange of three halos of width 1. In our current implementation, the halos are sent separately, so TMR creates three times the number of messages. While this could be optimized, we do not expect this to have a significant impact on performance for the (large) problem sizes of interest. For such problems, we have found that our advection solver (1) is primarily memory bound and secondly communication volume bound, and (2) scales at least to 2000 MPI processes [11]. We also use MPI Isend and Irecv to send messages, which minimizes the impact of the extra message startup times. Thus, we do not expect our implementation of TMR and robust stencils to have significant differences with respect to scalability.

In order to simulate memory corruption, we created a fault generator thread per each MPI process, which runs independently from the main computation thread. The thread flips a bit in the advection solver data area according to a given error probability at a fixed rate. This probability is proportional to the size of the area, so TMR has approximately three times the probability. This is basically the scheme used in [9].

4 Results

All experiments were conducted on the Raijin cluster managed by the National Computational Infrastructure (NCI) with the support of the Australian Government. Raijin has a total of 57,472 cores distributed across 3,592 compute nodes each consisting of dual 8-core Intel Xeon (Sandy Bridge 2.6 GHz) processors (i.e. 16 cores) with Infiniband FDR interconnect, an total of $\approx$ 160 terabytes of main memory, with an x86_64 GNU/Linux OS. `g++ -O3` is used to compute all results.

Figure 4 gives the elapsed time of all robust stencil combinations and TMR. 'NN' means S_{NN} etc (see Section 2). The other codes are as follows:

code	combination	code	combination
C30	WF, FW, FF	C50	NN, WW, WF, FW, FF
C31	WW, WF, FW		
C32	NN, NW, NF	TMR	NN $\times 3$

An (i, j) grid means the advection field size is $2^i \times 2^j$.

We see that each single stencil has similar performance, indicating that the computation is memory bound. We see that C30 and C31 slightly out-performs TMR, especially for when SSE2 is used, although SSE2 gives little or no benefit in any case. The extra cost of the 5

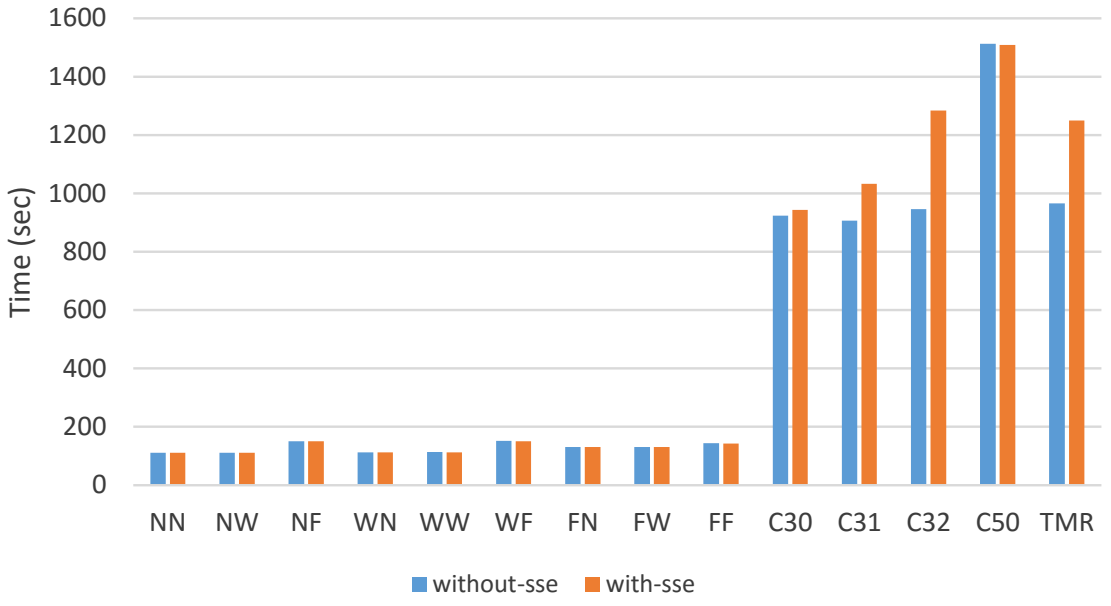

Figure 4: Elapsed time of each stencil, robust stencil combination, TMR (a (14, 14) grid and 512 steps, 16 MPI processes on 16 core Xeon processor).

stencil combination results in it being slower than TMR. Results with a smaller $(12, 12)$ grid were qualitatively similar, except SSE2 results were closer to the others.

Note that the results are with unoptimized codes.

The error in the final field is shown in Figure 5. An exact analytical solution is computed to determine the accuracy of our solver, with an L1-norm being employed. Again, results on the smaller $(12, 12)$ grid are qualitatively similar.

Figure 6 indicates the tolerance for each combination. The figure is best read looking right across a particular level of error in the solution. As expected, looking at where the injected error rate causes the computation to break down (Inf), the combinations become more robust the more stencils are added. Of the 3-stencil combinations, C31 tends to be slightly more robust. This is evident by looking along the 10^{-7} and 10^{-6} lines in the figure.

5 Conclusions

For 2D PDE solvers, robust stencils may be derived from various combinations of widened base stencils. These combinations permit the avoidance of any single corrupted point; hence the techniques can make the solver robust to soft faults. 3–5 stencil combinations are comparable to TMR in terms of robustness.

Robust stencils have one third of the memory footprint of TMR. In our current naive implementation, some 3-stencil combinations are slightly faster than TMR. We expect that, however, with suitable optimization, this difference will become dramatic.

Future work includes exploring optimizing the robust stencils, and exploring higher dimensions. For 3D, we would expect a set of 3 stencils (e.g. WWF, WFW and FWW; or NWF, WFN, FNW) would suffice. As only the F stencil involves extra points, we would only expect a 4:3 increase in terms of floating point operations over TMR. We also envisage GPU

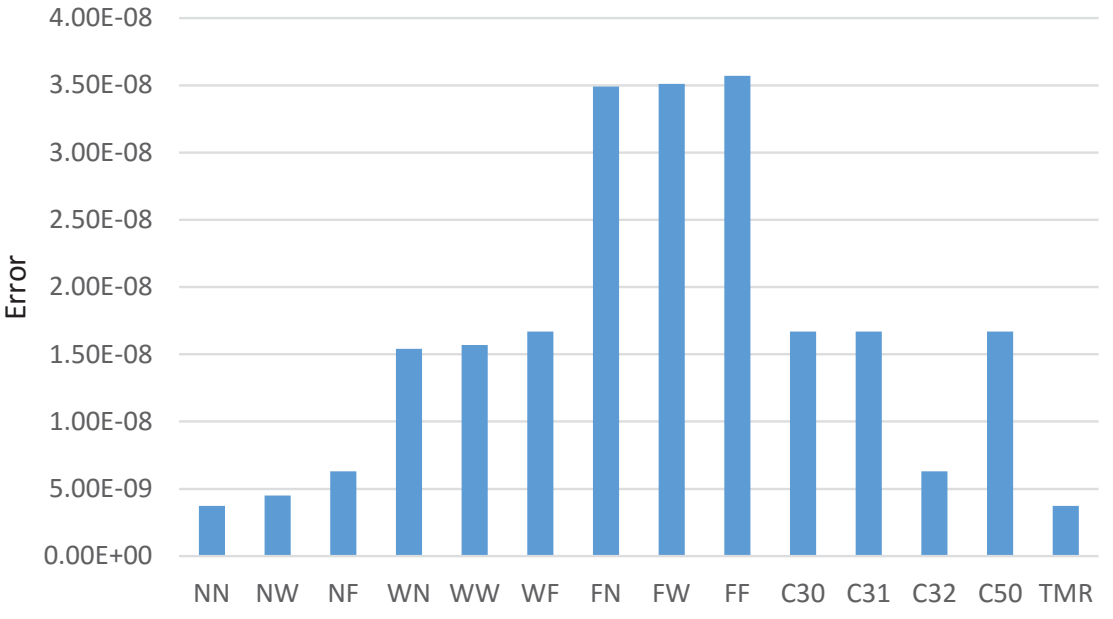

Figure 5: Error of final field of each stencil, robust stencil combination and TMR without fault injection (a (14, 14) grid, 512 steps, 16 MPI processes on 16 core Xeon processor).

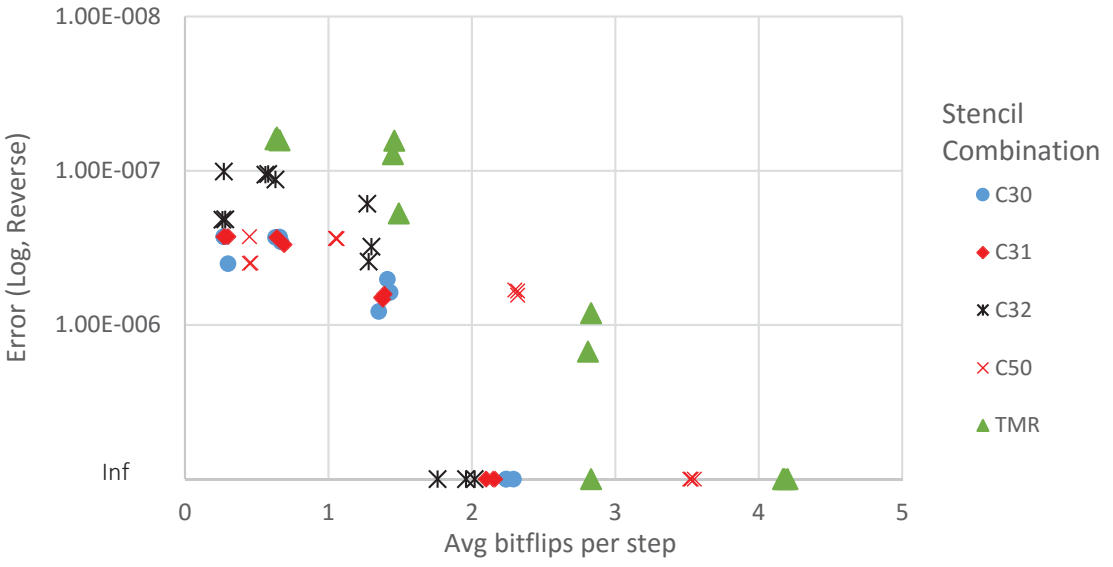

Figure 6: Error of final field of robust stencil combinations and TMR, according to the average number of bit flips per step with a (12, 12) grid and 128 steps (8 MPI processes + 8 memory corrupters on a 16 core Xeon processor). The Y-axis is reversed and the bottom line is Inf (higher is better).

implementations and generalizations to other PDEs.

5.1 Acknowledgments

We thank Markus Hegland for advice, and acknowledge financial support from ARC Linkage Project LP110200410. We thank NCI NF for the use of the Raijin cluster under project v29. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

[1] Md Mohsin Ali, James Southern, Peter E Strazdins, and Brendan Harding. Application level fault recovery: Using Fault-Tolerant Open MPI in a PDE solver. In *IEEE 28th International Parallel & Distributed Processing Symposium Workshops (IPDPSW 2014)*, pages 1169–1178, Phoenix, USA, May 2014.

[2] Md Mohsin Ali, Peter E Strazdins, Brendan Harding, Markus Hegland, and Jay W Larson. A fault-tolerant gyrokinetic plasma application using the sparse grid combination technique. In *Proceedings of the 2015 International Conference on High Performance Computing & Simulation (HPCS 2015)*, pages 499–507, Amsterdam, The Netherlands, July 2015.

[3] Franck Cappello. Fault tolerance in petascale/ exascale systems: Current knowledge, challenges and research opportunities. *International Journal of High Performance Computing Applications*, 23(3):212–226, 2009.

[4] Franck Cappello, Al Geist, Bill Gropp, Laxmikant Kale, Bill Kramer, and Mark Snir. Toward exascale resilience. *International Journal of High Performance Computing Applications*, 2009.

[5] Franck Cappello, Al Geist, William Gropp, Sanjay Kale, Bill Kramer, and Marc Snir. Toward exascale resilience: 2014 update. *Supercomputing Frontiers and Innovations*, 1(1):5–28, 2014.

[6] Christian Engelmann, Hong H. Ong, and Stephen L. Scott. The case for modular redundancy in large-scale high performance computing systems. In *Proceedings of the 8th IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN) 2009*, pages 189–194, Innsbruck, Austria, February 16-18, 2009. ACTA Press, Calgary, AB, Canada.

[7] W. Hundsdorfer and J.G. Verwer. *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*. Springer Series in Computational Mathematics. Springer, 2003.

[8] Peter Lax and Burton Wendroff. Systems of conservation laws. *Communications on Pure and Applied Mathematics*, 13(2):217–237, 1960.

[9] Jaideep Ray, Jackson Mayo, and Robert Armstrong. Finite difference stencils robust to silent data corruption. In *SIAM Conference on Parallel Processing*, Feb 2014.

[10] Marc Snir, Robert W. Wisniewski, Jacob A. Abraham, Sarita V. Adve, Saurabh Bagchi, Pavan Balaji, Jim Belak, Pradip Bose, Franck Cappello, Bill Carlson, Andrew A. Chien, Paul Coteus, Nathan DeBardeleben, Pedro C. Diniz, Christian Engelmann, Mattan Erez, Saverio Fazzari, Al Geist, Rinku Gupta, Fred Johnson, Sriram Krishnamoorthy, Sven Leyffer, Dean Liberty, Subhasish Mitra, Todd Munson, Rob Schreiber, Jon Stearley, and Eric Van Hensbergen. Addressing failures in exascale computing. *International Journal of High Performance Computing Applications (IJHPCA)*, 28(2):129–173, 2014.

[11] Peter E Strazdins, Md Mohsin Ali, and Brendan Harding. Highly scalable algorithms for the sparse grid combination technique. In *Proceedings of the IEEE 29th International Parallel & Distrib uted Processing Symposium Workshops (IPDPSW 2015)*, pages 941–950, Hyderabad, India, May 2015.