Two Approaches to Highly Scalable and Resilient Partial Differential Equation Solvers

Peter Strazdins
Computer Systems Group,
Research School of Computer Science,
The Australian National University;

(slides available from http://cs.anu.edu.au/~Peter.Strazdins/seminars)

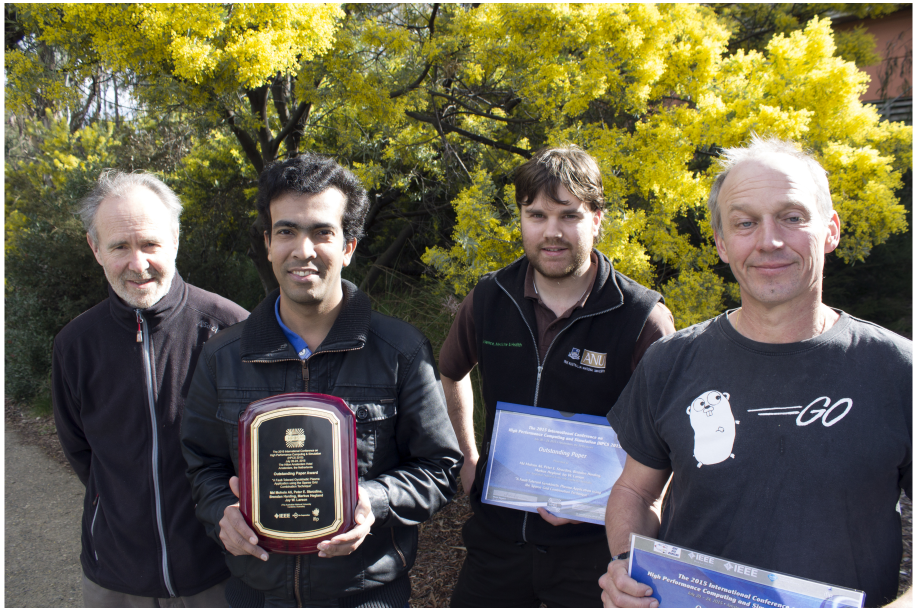
Co-Lab Seminar, Mathematical Sciences Institute & The Research School of Computer Science, The Australian National University, 18 October 2019

THE AUSTRALIAN NATIONAL UNIVERSITY

1 Context of Talk: Parent Project

LP11 *Robust numerical solution of PDEs on petascale computer systems with applications to tsunami modelling and plasma physics* (Hegland et al)

- large-scale parallelization of the ANUGA tsunami application
- (hard) fault tolerant computation of PDEs with the Sparse Grid Combination Technique (SGCT)
- highly scalable parallel SGCT algorithms
- complex parallel applications made fault tolerant via the SGCT
  - GENE (plasma physics); also Taxilla Lattice-Boltzmann and Solid Fuel Ignition
  - hard faults; assumes constant resources (replace failed processes)



2 Overall Organization of Talk

This talk is about follow-on topics, organized in two parts:

1. PDE application-level (hard) fault tolerance for shrinking resources (continue without failed processes) via the SGCT

   (joint work with Mohsin Ali (then ANU) and Bert Debusschere (Sandia National Laboratories))

2. robust stencils as a general method to deal with soft faults in PDE solvers

   (joint work with Brendan Harding & Brian Lee (then ANU), and Jackson Mayo, Jaideep Ray, Robert Armstrong (Sandia National Laboratories))

Common theme throughout: advection as an example PDE solver.

3 Part 1 Overview: FT for Shrinking Resources via the SGCT

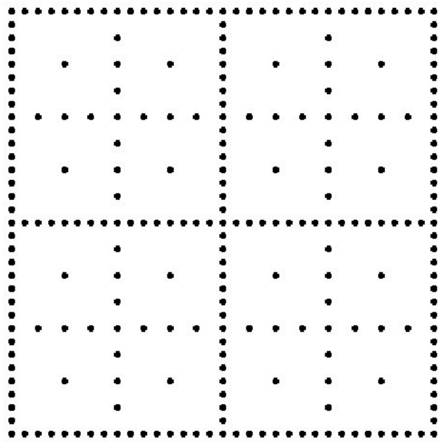
- motivation: why make applications fault-tolerant?
- background:
  - solving PDEs via sparse grids with the combination technique
  - the robust combination technique
  - parallel sparse grid combination technique (SGCT) algorithm overview
- shrinkage-based recovery from faults
- fault detection and recovery using ULFM MPI (Message Passing Interface)
- SGCT algorithm support for shrinkage
- modifications to the application (a 2D advection PDE solver)
- results: comparison with process replacement and checkpointing, performance and accuracy
- conclusions and future work

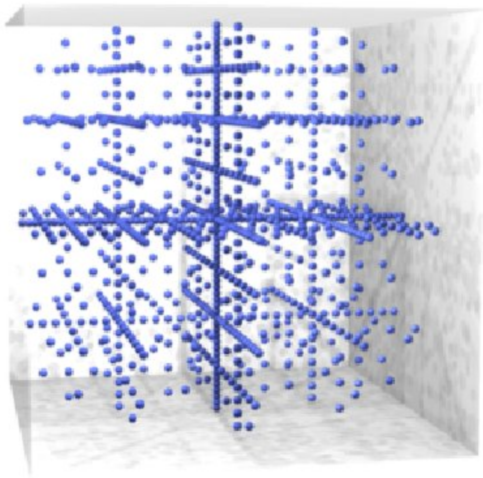
4 Motivation: Why Fault-Tolerance is Becoming Important

- exascale computing: for a system with n components, the mean time before failure is proportional to $1/n$
  - a sufficiently long-running application will *never* finish!
  - by 'failure' we usually mean a transient or permanent failure of a component (e.g. node) – this is called a hard fault
- cloud computing: resources (e.g. compute nodes) may have periods of scarcity / high costs
  - for a long-running application, may wish to shrink and grow the nodes it is running on accordingly – this scenario is also known as elasticity
- low power or adverse operating condition scenarios may cause failures even with a moderate number of components
  - this typically results in corrupted data – a soft fault
- the SGCT is a form of algorithm-based fault tolerance capable of meeting these challenges for a range of scientific simulations

5 Background: Sparse Grids

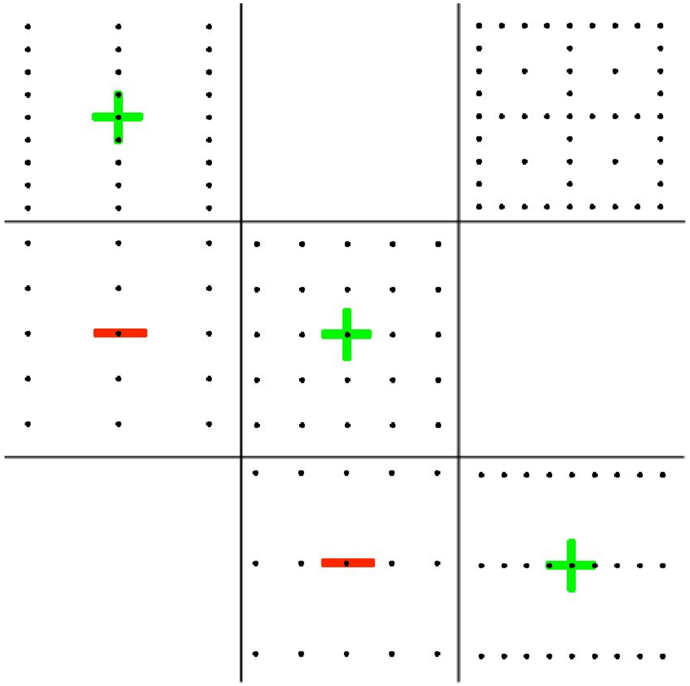
- introduced by Zenger (1991)
- for (regular) grids of dimension d having uniform resolution n in all dimensions, the number of grid points is n^d
  - known as the *curse of dimensionality*
- a sparse grid provides fine-scale resolution
- can be constructed from regular sub-grids that are fine-scale in some dimensions and coarse in others
- has been proved successful for a variety of different problems:
  - good accuracy for given effort ($O(n \lg(n)^{d-1})$ points)
  - various options for fault-tolerance!

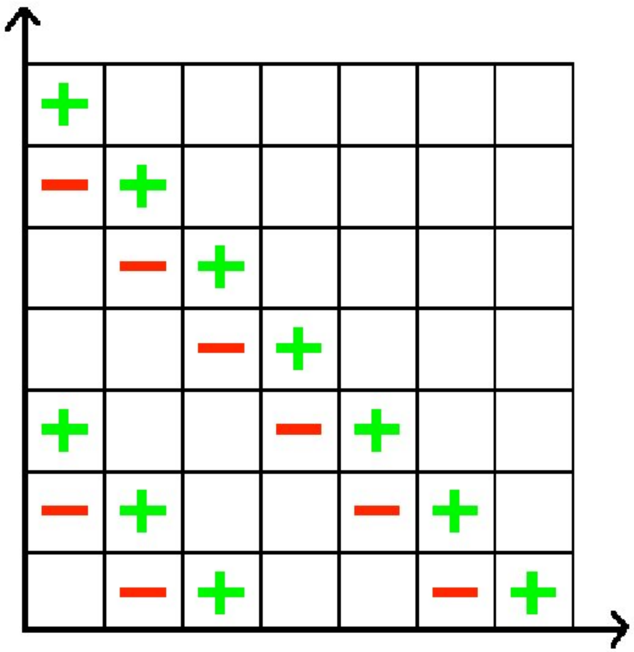




6 Background: Combination Technique for Sparse Grids

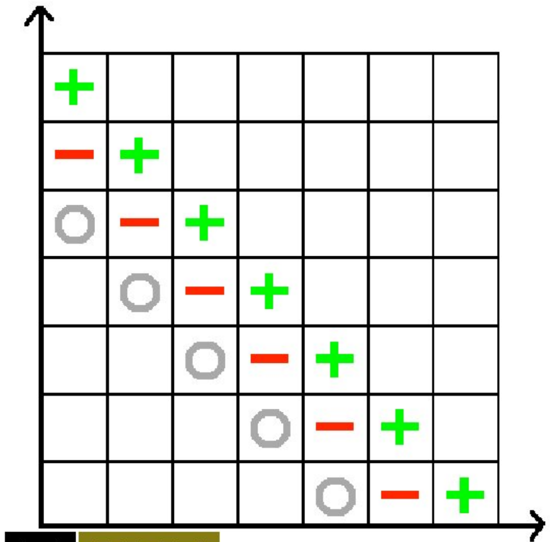
- computations over sparse grids may be approximated by being solved over the corresponding set of regular sub-grids
  - overall solution is from 'combining' sub-solutions via an inclusion-exclusion principle (complexity is still $O(n\lg(n)^{d-1})$ where $n = 2^l + 1$)
- for 2D at 'level' $l = 3$, combine grids $(3,1)$, $(2,2)$ $(1,3)$ minus $(2,1)$, $(1,2)$ onto (sparse) grid $(3,3)$ (interpolation is required)

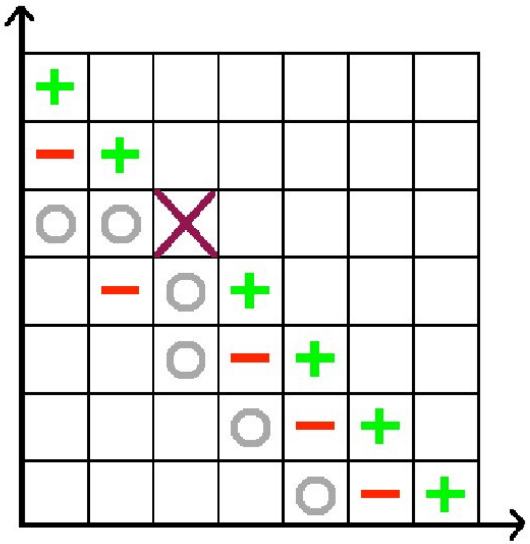




7 Robust Combination Techniques

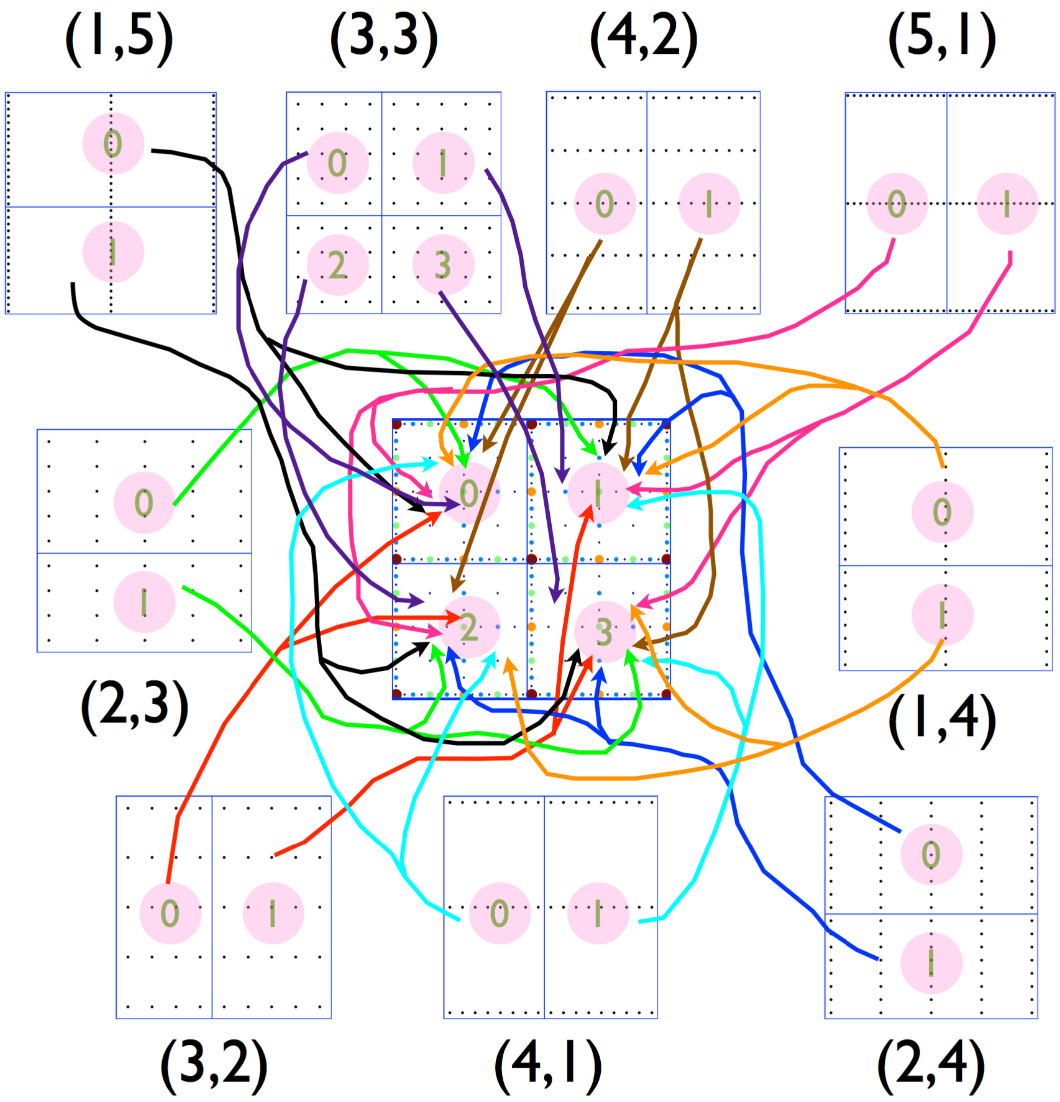
- uses extra set of smaller sub-grids
  - the redundancy from this is $< 1/(2(2^d - 1))$
- for a single failure on a sub-grid, can find a new combination formula with an inclusion/exclusion principle avoiding the failed sub-grid
- works for many cases of multiple failures (using a 4th set covers all)
- a failed sub-grid can be recovered from its projection on the combined sparse grid





THE AUSTRALIAN NATIONAL UNIVERSITY

8 Parallel SGCT Algorithm: the Gather-Scatter Idea



- evolve independent simulations over time T on a set of component grids, solution is a d-dimensional field (here $d{=}2, l{=}5$)
- each grid is distributed over a *process grid* (here these are 2×2, 2×1 or 1×2)
- gather: combine fields on a sparse grid (index $(5,5)$), here on a 2×2 process grid
- scatter: sample (the more accurate) combined field and redistribute back to the component grids

9 Shrinkage-based Recovery of FT SGCT Applications

- each sub-grid is solved over a set of processes (with contiguous MPI ranks within the global MPI communicator)
- we check for process failure before applying the SGCT
- after detection of failure, the faulty communicator is shrunk, containing only the alive processes
- we shrink the process sets of the sub-grids that experienced the failures
  - we have also to shrink the local sizes of the sub-grids and associated data structures in these processes!

  This seems hard! However:
  - FT apps generally must be capable of a restart from the middle; similarly we can implement a 're-size'
  - the FT-SGCT provides an effectively cost and effort-free redistribution!
- processes of other sub-grids merely get their ranks adjusted

10 Shrinkage-based Recovery of an $l = 4$ FT SGCT Application

24-31			
40-43	16-23		
46-47	36-39	8-15	
48	44-45	32-35	0-7

(a) process sets before shrinking communicator

0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7

~~8~~	9	10	11	12	13	14	15
-	8	9	10	11	12	13	14

16	17	18	19	20	21	22	23
15	16	17	18	19	20	21	22

24	25	26	27	28	29	30	31
23	24	25	26	27	28	29	30

32	33	34	35
31	32	33	34

36	37	~~38~~	39
35	36	-	37

40	41	42	43
38	39	40	41

44	45
42	43

46	47
44	45

48
46

(b) details before & after

23-30			
38-41	15-22		
44-45	35-37	8-14	
46	42-43	31-34	0-7

(c) process sets after shrinking communicator

11 Communicator Recovery via ULFM MPI

- recovery via process shrinkage similar to process replacement
- create an ULFM MPI error handler, passing address of the global communicator `ftComm` to it
- e.g. processes 3 and 5 of ranks 0–6 now fail

  0 1 2 3 4 5 6
- before invoking the SGCT, call `MPI_Barrier(ftComm)` (this will now fail)

  0 1 2 4 6
- call `OMPI_Comm_revoke(&ftComm)`, create a shrunken communicator via `OMPI_Comm_shrink(ftComm, &shrunkComm)`

  0 1 2 3 4
- synchronize the system via `OMPI_Comm_agree(ftComm=shrunkComm, ...)`
- note: must reset any local variables dependent on the MPI rank or communicator size

12 SGCT Algorithm Support for Shrinkage

- for a 2D SGCT-enabled application, each sub-grid is decomposed over a subset of MPI processes arranged as a 2D *process grid*, containing:
  - n, the total number of processes available
  - r_0, the MPI rank of the first process
  - $P = (P_x, P_y)$, the process grid shape. Initially $n = P_x P_y$

  A logical process id $p = (p_x, p_y)$, $(0, 0) \leq p < P$, has rank $r_0 + p_y P_x + p_x$
- if this grid is numbered $i \geq 0$, $r_0 = \Sigma_{j=0}^{i-1} n_j$, where n_j is number of processes in grid j
- if we detect f failures in this grid, we resize to $P \leftarrow (P_x - \lceil f/P_y \rceil, P_y)$ and set $n \leftarrow n - f$
- if we detect f_l failures in process grids to left (numbered $j < i$), $r_0 \leftarrow r_0 - f_l$

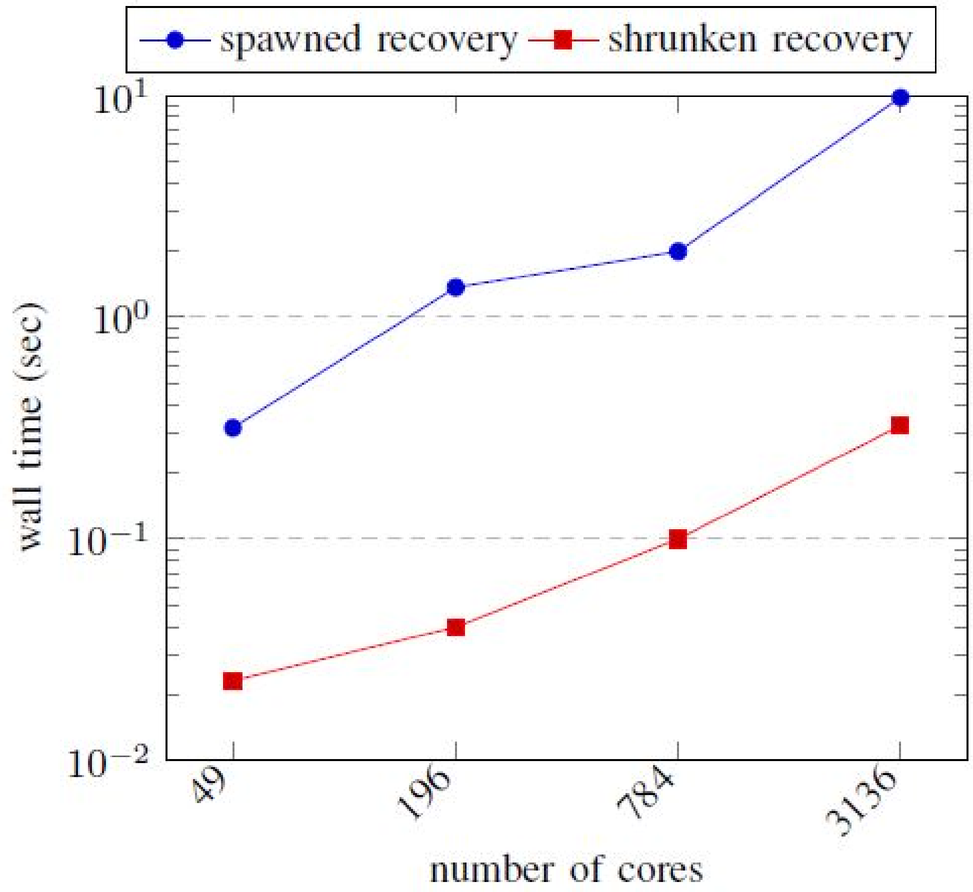
13 Modifications to the PDE Solver

- the initialization of all process grid dependent variables and arrays are put into a single function
  - note that an FT application (e.g. by checkpointing) will have to do this as well, to facilitate restart at an arbitrary point
- before calling the SGCT, a list of ranks of all failed processes is made
- if the current process grid has one of these, it does not participate in the *gather* stage of the SGCT
  - it however re-sizes its data, calling the initialization function
  - it participates in the *scatter* stage, receiving its re-sized solution field automatically

Otherwise, perform the *gather* and *scatter* of the SGCT as per normal

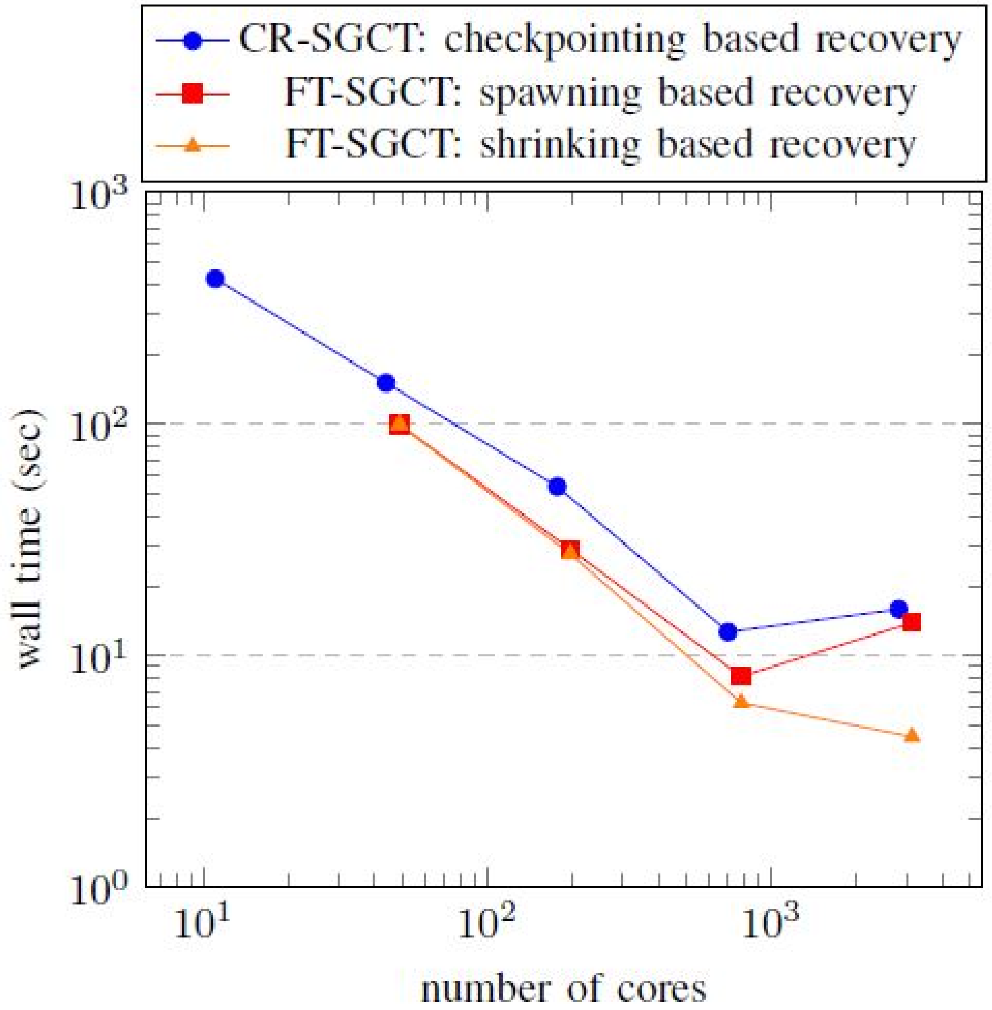
14 Results: Replace vs Shrink Recovery Overheads

- compare overheads of process replacement ('spawn') vs process shrinkage
- experiments on the Raijin cluster, dual 8-core Sandy Bridge 2.6 GHz nodes + Infiniband FDR
- uses ULFM's (slower) Two-Phase Commit distributed agreement algorithm
- 2 random process failures via `kill` signals



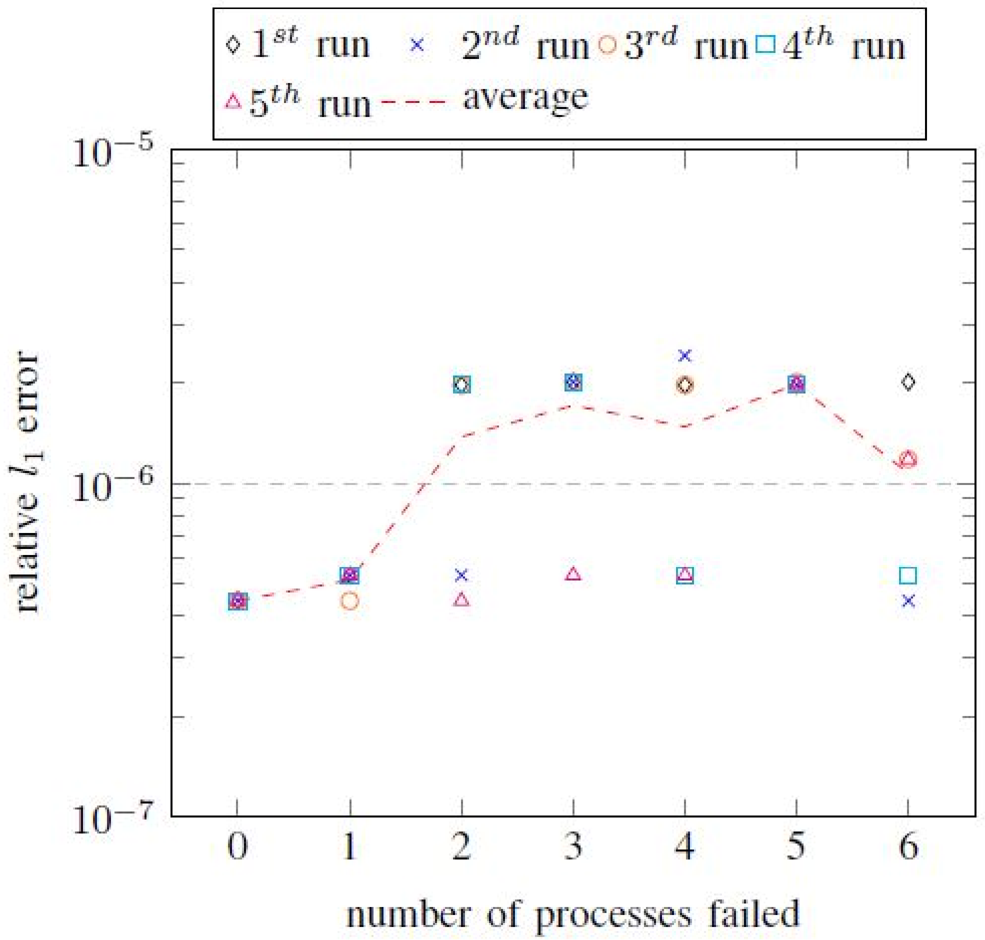
15 Results: Advection Application Performance

- compared also with a CR version of a 2D SGCT advection solver
- SGCT with level $l = 4$ over a $2^{13} \times 2^{13}$ (full) grid
- 2 random process failures: sufficient to reveal interesting recovery behavior
- ULFM agreement algorithm impacts on performance for ≈ 3000 cores
- shrinkage fastest despite loss of compute resources



16 Results: Advection Application Accuracy

- FT SGCT with level $l = 4$ over a $2^{13} \times 2^{13}$ (full) grid
- random process failures over initial set of 49 processes
- baseline error rate (no failures) is 4.45E-07
- error for each test depend on which sub-grids had the failed processes
- note: results identical for replacement or shrinkage recovery

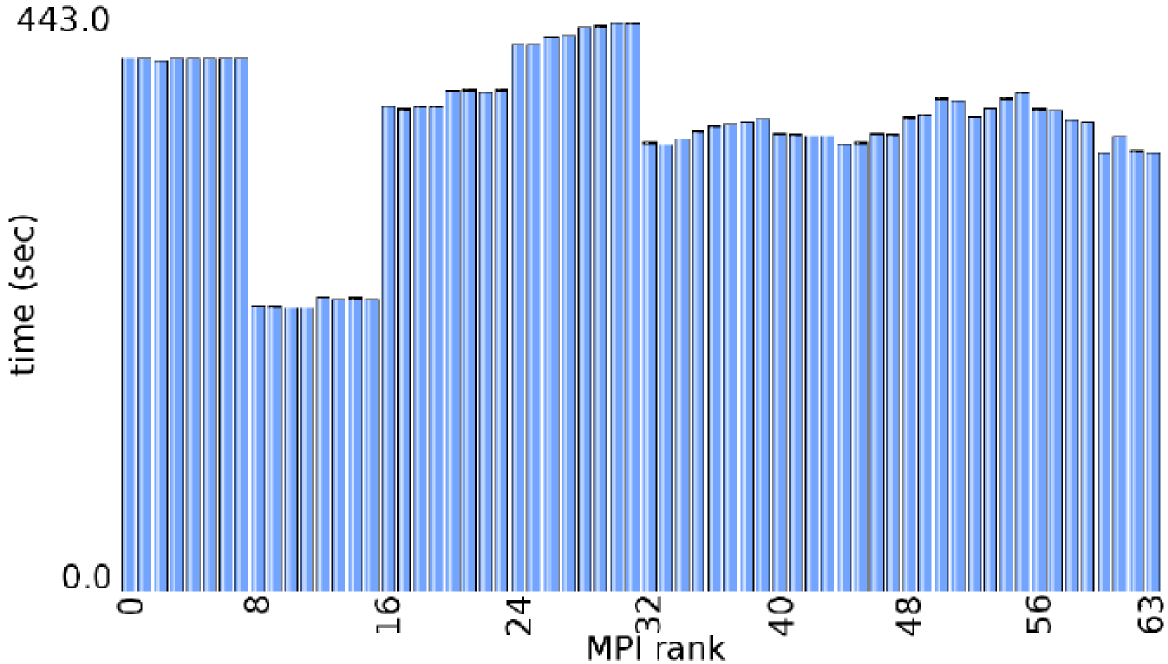


17 Part 1: Conclusions

- demonstrated that SGCT applications can be made fault tolerant under a shrinkage regime
- recovery under ULFM MPI is relatively simple and reliable
  - also order of magnitude faster than the replacement regime
- existing parallel SGCT algorithm needed only process grid re-sizing support added
  - the SGCT automatically solves the problem of redistribution!
- only modest modifications on an existing FT application is required
- with small numbers of failures, shrinkage gave faster application performance than replacement (and $\approx 2\times$ faster checkpoint-restart)
  - would improve with a more scalable ULFM distributed agreement algorithm
- advection solver accuracy still high even with $\approx 10\%$ process failures

18 Part 1: Future Work

- extend for elasticity: growing as well as shrinking resources
- extend to real applications, e.g. the GENE gyrokinetic plasma application
  - no in-principle reason why not, especially as a GENE is already restartable (from checkpoints)



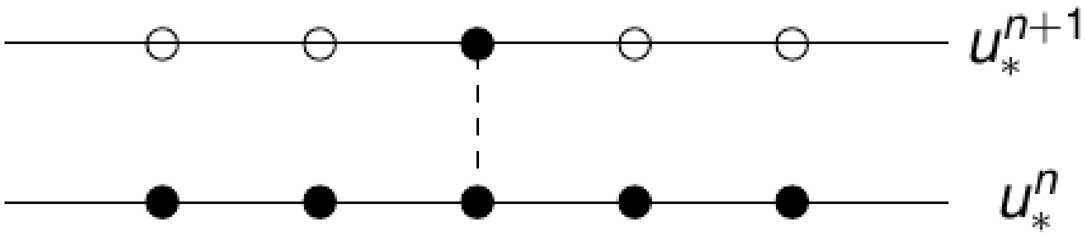
19 Part 2: Robust Stencils – Motivations for Soft Faults

- soft or silent faults also have exposure/risk increasing with system size or reduced power levels
- generic solutions: triple modular redundancy (TMR), checkpoint-restart
- active research area in recent decades
- various papers discuss the use of checksums to detect and correct memory failures (bit flips) in linear algebra
- we present an approach for avoiding bit flip errors in finite difference computations

(some text and diagrams for this part are borrowed from Brendan Harding's ICCS'16 slides)

20 Finite Difference Computations

- finite difference methods are common for (explicitly) solving partial differential equations



- explicit methods cannot leverage fault tolerant linear algebra techniques
- triple modular redundancy (TMR) could easily be used
  - do everything 3 times (with separate memory)
  - choose the result which is equal for any two
  - 1/3 efficiency (3 times the memory and time)
- how else could we detect/correct errors?

21 Robust Stencils in 1D

- the 1D advection equation $\delta_t u + a\delta_x u = 0$
  may be solved by the standard ('normal') Lax-Wendroff method:

$$u_t^{n+1} = \frac{c(1+c)}{2}u_{i-1}^n + (1-c^2)u^{n-1} + \frac{c(-1+c)}{2}u_{i+1}^n$$

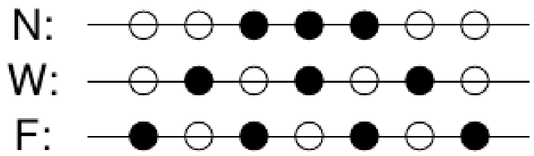
  where $c = a\Delta t/\Delta x$, and is stable and of second order
- Mayo et al. used several finite difference discretisations for fault tolerance, which are also stable and of second order
  - the widened discretisation, avoiding the $i \pm 1$ points:

$$u_i^{n+1} = \frac{c(2+c)}{8}u_{i-2}^n + \frac{4-c^2}{4}u_i^n + \frac{c(-2+c)}{8}u_{i-2}^n$$

  - the third (far) discretisation, avoiding the central point:

$$u_i^{n+1} = \frac{-3+8c+3c^2}{48}u_{i-3}^n + \frac{9-c^2}{25}u_{i-1}^n + \frac{9-c^2}{25}u_{i+1}^n + \frac{-3-8c+3c^2}{48}u_{i+3}^n$$

- the corresponding stencils are:



22 Advection in 2 or More Dimensions

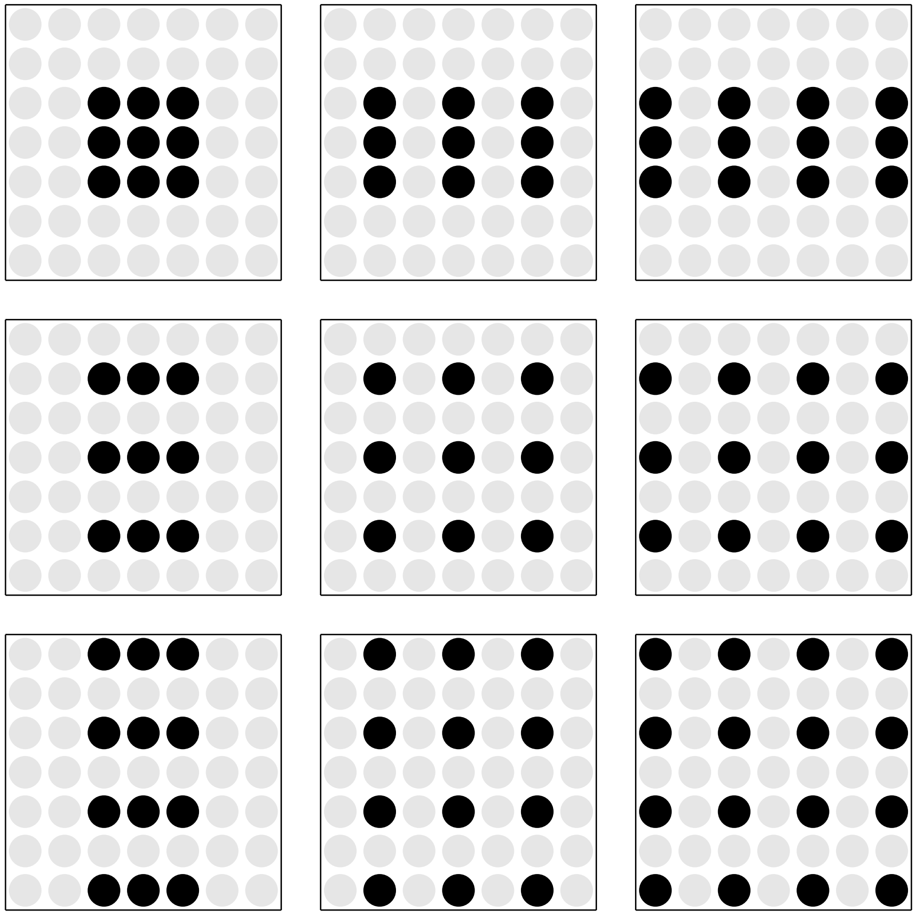
- wish to extend to the 2D advection equation:

  $\delta_t u + a\delta_x u + b\delta_y u = 0$

- assume a square domain discretised as a uniform grid
- using the N, W and F stencils, the tensor product of the coefficients in the x- and y- dimensions gives the 2D coefficients
- the 3×3 resulting stencils are

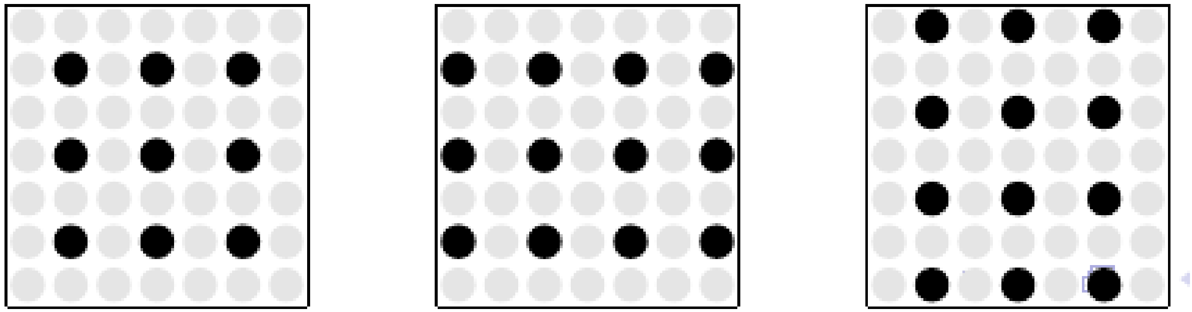
NN	WN	FN
NW	WW	FW
NF	WF	FF

- this approach can be generalized to $d > 2$



23 Robust Stencils in 2 Dimensions

- under the assumption of a single faulty point in the 7×7 region, how do we choose a stencil to avoid that point?
  - preferably in an application-independent fashion
- idea: for each point, compute a subset of the 9 stencils and take the median as the result
  - chose subsets of $s = 3, 5, 7$ stencils so that no one point is in any more than $(s-1)/2$ of them, then the stencil with the median will not contain any one faulty point



24 2D Robust Stencil Sets

- ideally, we want the most accurate stencil (NN) in the set
  - *hopefully* the other stencils will bracket this in error-free regions
- prefer to use symmetric stencil sets
- with the condition that no one point is in $(s-1)/2$ of them, there is only one of these, having $s = 5$
- robust $s = 3$ and $s = 7$ sets (not including NN) are also shown below:

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

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

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

THE AUSTRALIAN NATIONAL UNIVERSITY

25 Analysis Relative to Standard Lax-Wendroff

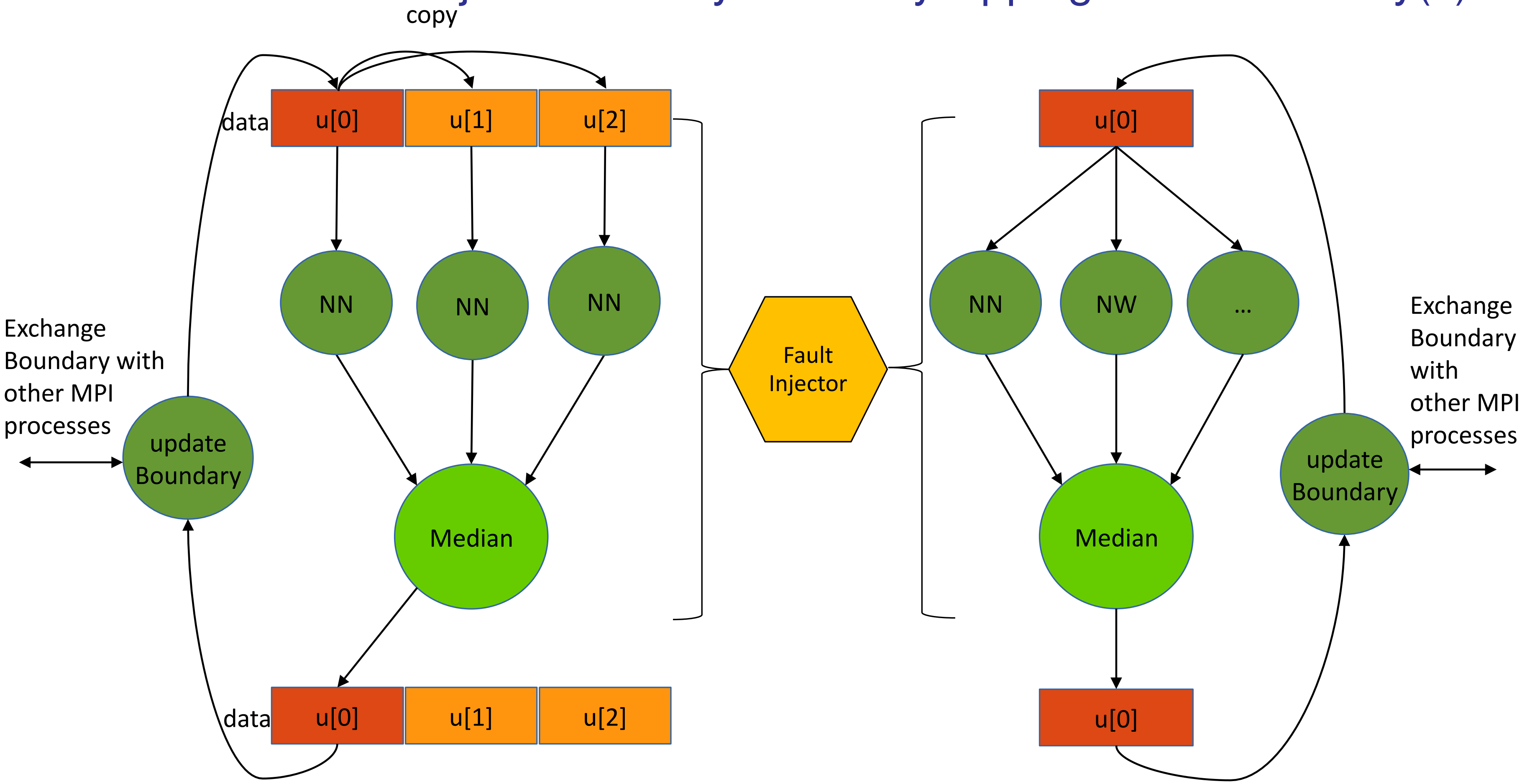
(TMR = triple modular redundancy)

	TMR	robust stencils (3/5/7 sets)
memory	$3\times$	$\approx 1\times$
FLOPs	$3\times$ (plus median)	$3.7/6.4/8.3\times$ (plus median)
communication	$3\times$	$\approx 3\times$ (wider halos)
robustness, for 1 fault	yes	yes
robustness, for 2 faults (if not within a region of)	$3 \times 3 \cdot 2 \cdot 3 = 56$	$7 \times 7 = 49$

Note: stencil computations are typically memory bound, FLOPs may not reflect execution time, and TMR may have more cache misses.

26 Fault Injection

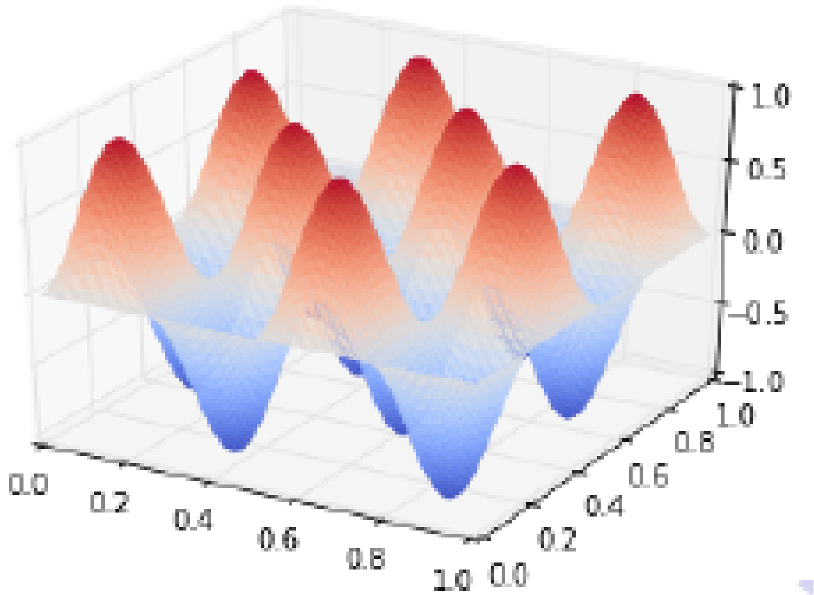
An additional thread injects faults by randomly flipping bits in the array(s).



There is an exponentially distributed fault injection rate.

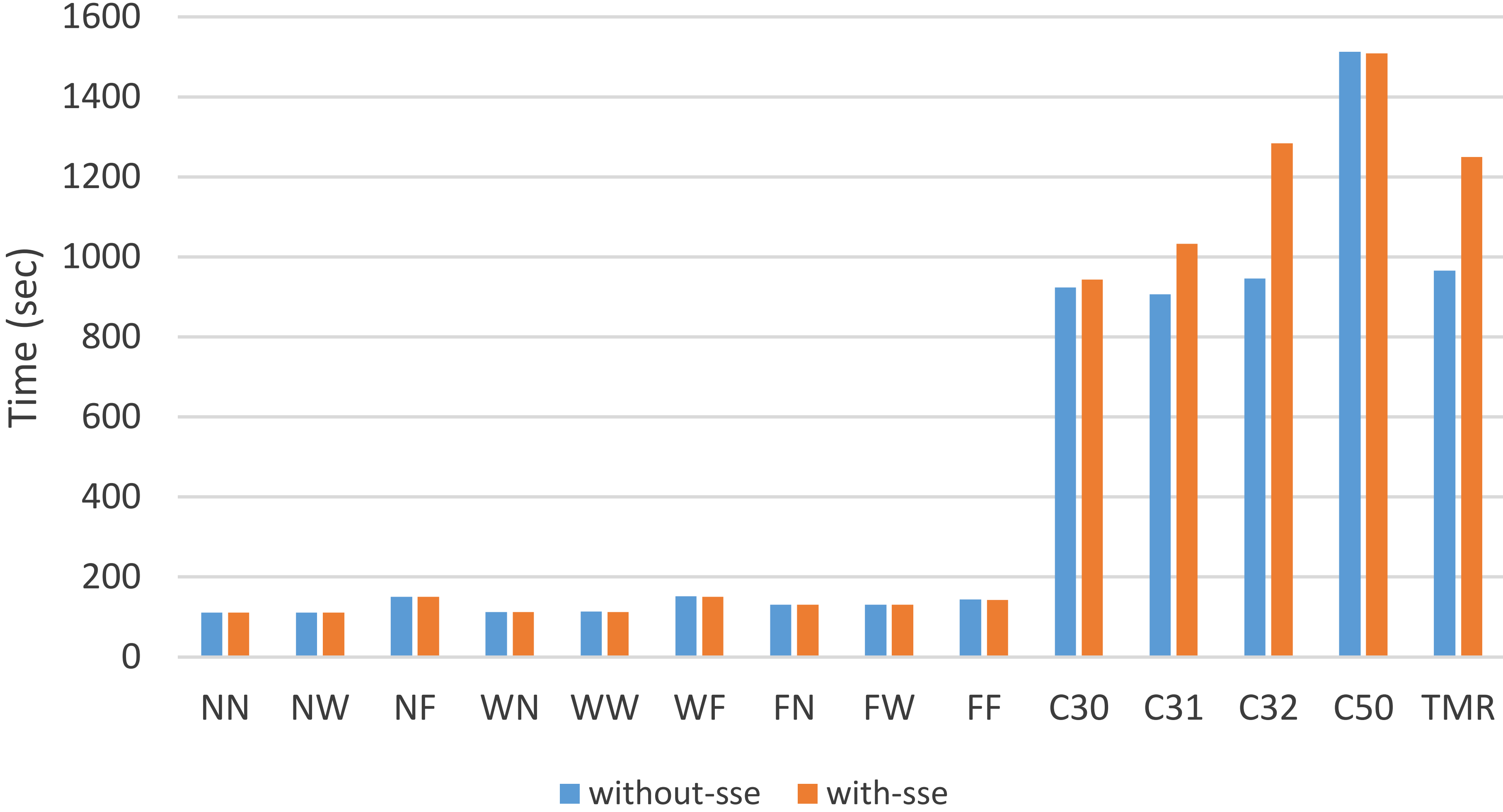
27 Other Implementation Details

- codes parallelized with MPI with Isend/Irecv for halos, scales to 2K cores
- codes were compiled on the NCI Raijin cluster with `mpic++ -O3`
- codes were not yet optimised
- initial condition is a sinusoidal field (4 peaks in x-dim, 2 in y-dim) over the unit square with uniform velocity (1.0,1.0)



28 Results – Execution Time

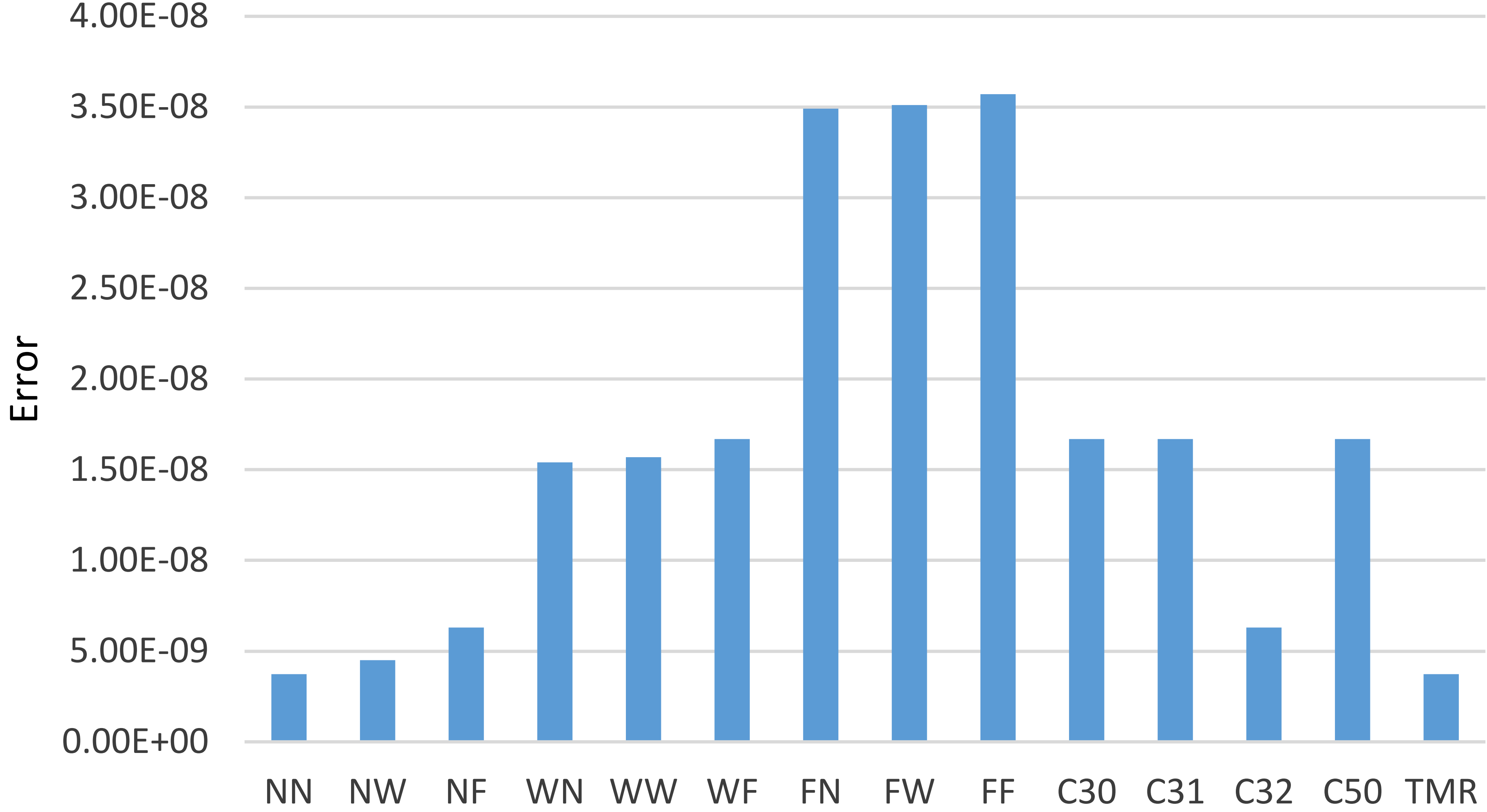
$2^{14} \times 2^{14}$ field, 512 timesteps, 4×4 MPI processes on a 16 core Xeon



(C30 = WF,FW,FF; C31=WW,WF,FW; C32=NN,NW,NF; C50=NN,WW,WF,FW,FF)

29 Results – Accuracy, Fault-free Case

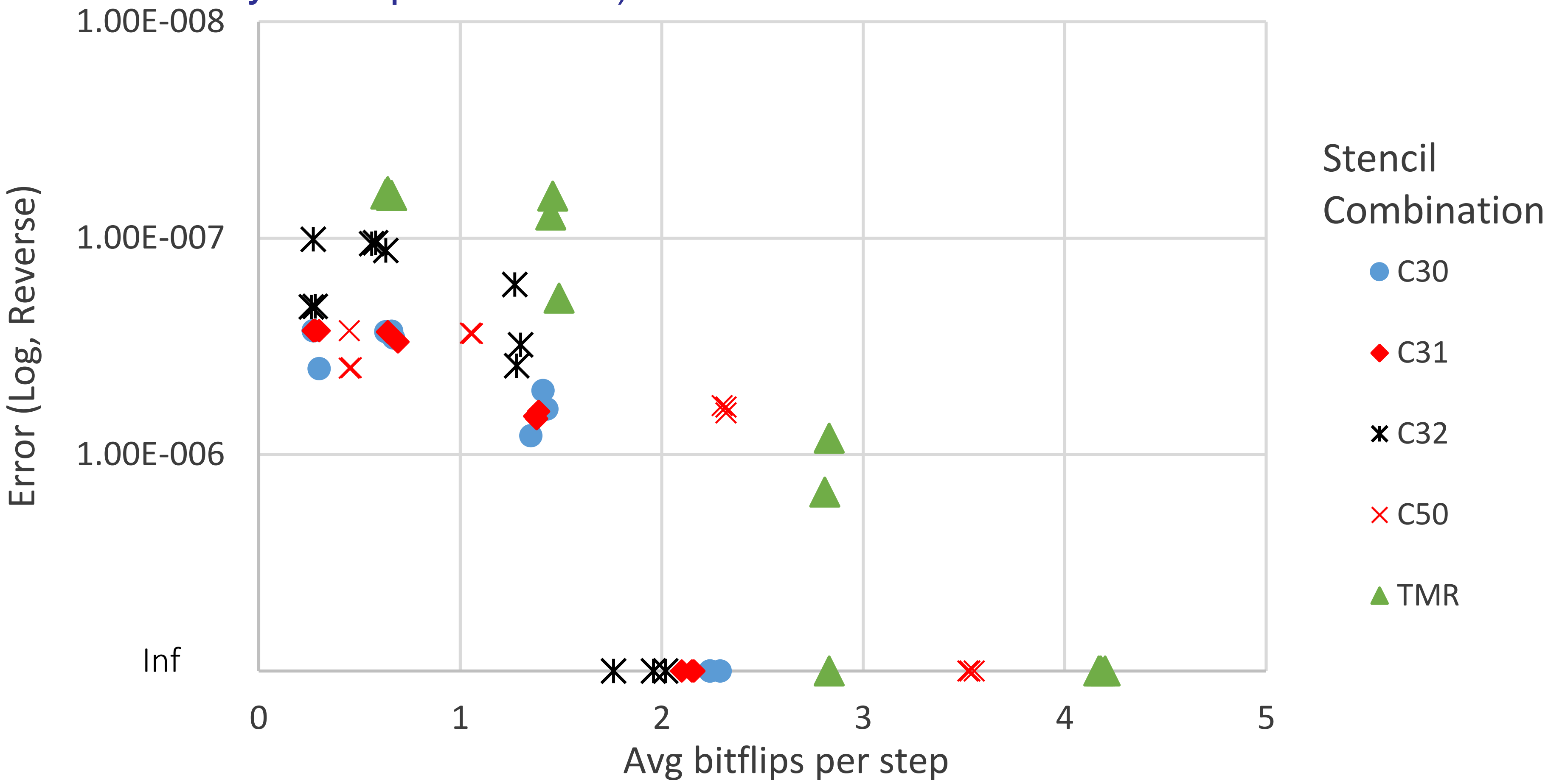
Average error for a $2^{14} \times 2^{14}$ field, 512 timesteps



(C30 = WF,FW,FF; C31=WW,WF,FW; C32=NN,NW,NF; C50=NN,WW,WF,FW,FF)

30 Results – Robustness

Average error for $2^{12} \times 2^{12}$ field, 128 timesteps, 8 MPI processes (each with a memory corruptor thread) on a 16 core Xeon



(C30 = WF,FW,FF; C31=WW,WF,FW; C32=NN,NW,NF; C50=NN,WW,WF,FW,FF)

31 Part 2: Conclusions

- robust stencils may be derived from various combinations of widened base stencils (this implies $\approx 4\times$ loss of accuracy)
- coefficients of 2D stencils are derived from the 'tensor product' of two 1D stencils
- application-independent selection of the 'best' stencil (via median)
- concepts can be extended to higher dimensions and/or other finite difference discretisations
- 3–5 stencil combinations are comparable to TMR in terms of robustness, comparable or better in terms of speed
- new work includes optimizations for stencil combinations, and
  - use 2 stencils at first to detect faults, engage more upon detection (needs application-dependent error threshold)
  - using stencil combinations / higher order stencils to improve accuracy in the fault-free case

32 Acknowledgements

- Markus Hegland for advice and support from the parent project funded by ARC Linkage Grant LP110200410
- use of Raijin from the National Computational Infrastructure (NCI) 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 Energys National Nuclear Security Administration under contract DE-AC04-94AL85000

. . . Questions???