Two Approaches to Highly Scalable and Resilient Partial Differential 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)

Lawrence Livermore National Laboratories Seminar, Livermore, USA, 18 August 2018

THE AUSTRALIAN NATIONAL UNIVERSITY

1 Overall Organization of Talk

1. Application Fault Tolerance for Shrinking Resources via the Sparse Grid Combination Technique (SGCT)

   (joint work with Mohsin Ali (RSCS ANU) and Bert Debusschere (Combustion Research Facility, Sandia National Laboratories))

   (a) extension to elastic (growing and shrinking) grids

   (b) possible extension of SGCT techniques to recover fro soft faults

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

   (joint work with Brendan Harding & Brian Lee (ANU), and Jackson Mayo, Jaideep Ray. Robert C. Armstrong (Robert Clay's group, Sandia National Laboratories)

2 Talk Overview: FT for Shrinking Resources via the SGCT (Part 1)

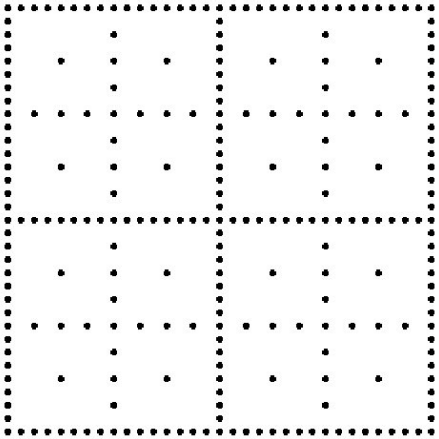
- motivation: why make applications fault-tolerant?
- background:
  - solving PDEs via sparse grids with the combination technique
  - the robust combination technique
  - parallel sparse grid combination technique (SGCT) algorithm overview
- shrinkage-based recovery from faults
- fault detection and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

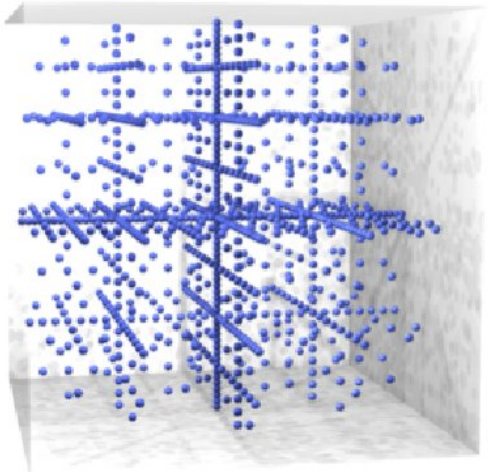
3 Motivation: Why Fault-Tolerance is Becoming Important

- 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 moderate number of components
- the SGCT is a form of algorithm-based fault tolerance capable of meeting these challenges for a range of scientific simulations

4 Background: Sparse Grids

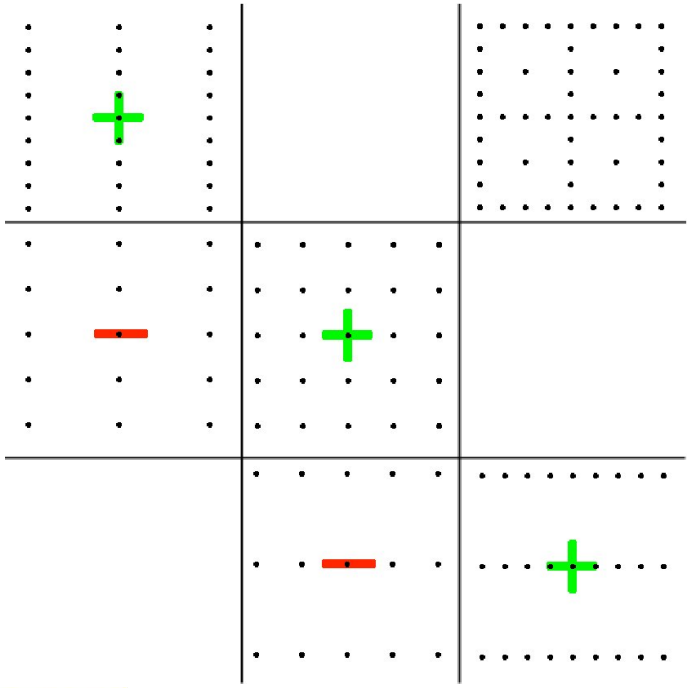
- introduced by Zenger (1991)
- for (regular) grids of dimension d having uniform resolution n in all dimensions, the number of grid points is n^d
  - known as the *curse of dimensionality*
- a sparse grid provides fine-scale resolution
- can be constructed from regular sub-grids that are fine-scale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

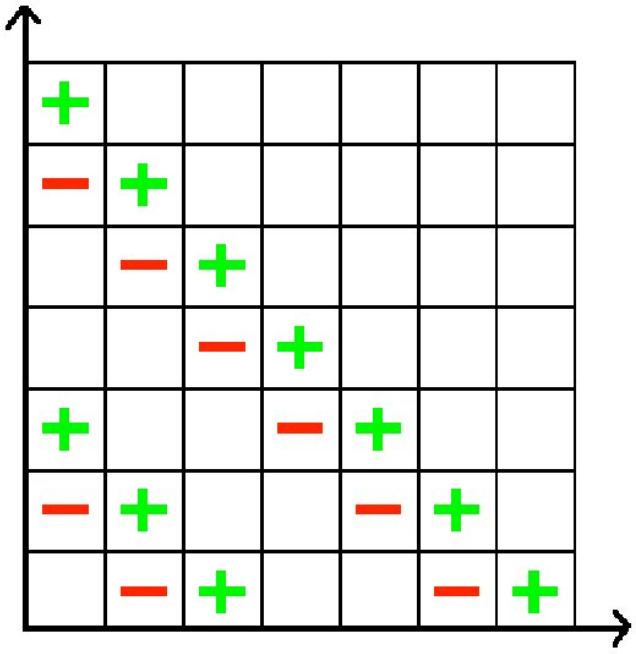




5 Background: Combination Technique for Sparse Grids

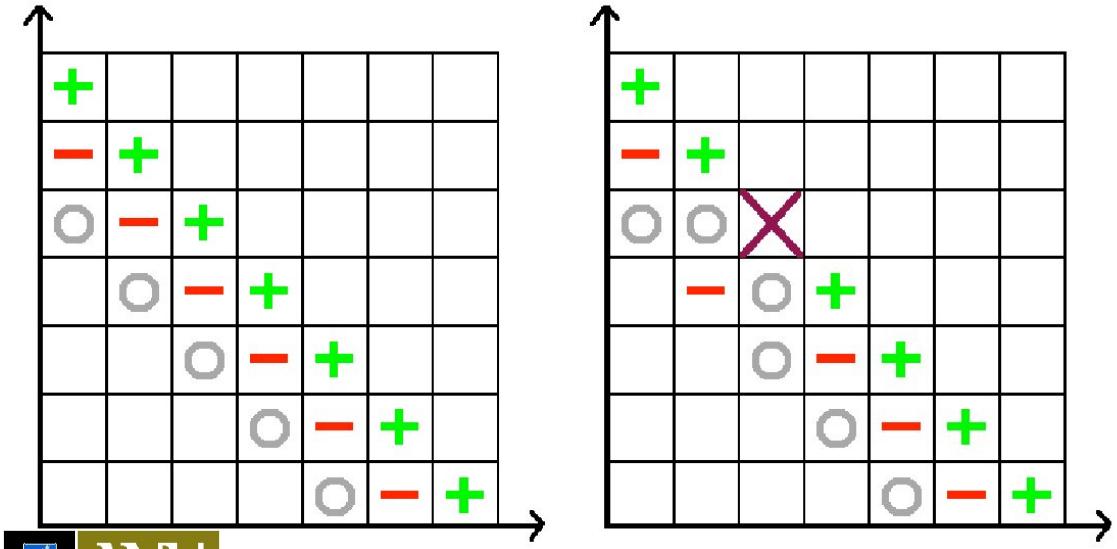
- computations over sparse grids may be approximated by being solved over the corresponding set of regular sub-grids
  - overall solution is from 'combining' sub-solutions via an inclusion-exclusion principle (complexity is still $O(n \lg(n)^{d-1})$)
- for 2D at 'level' $l = 3$, combine grids $(3,1)$,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



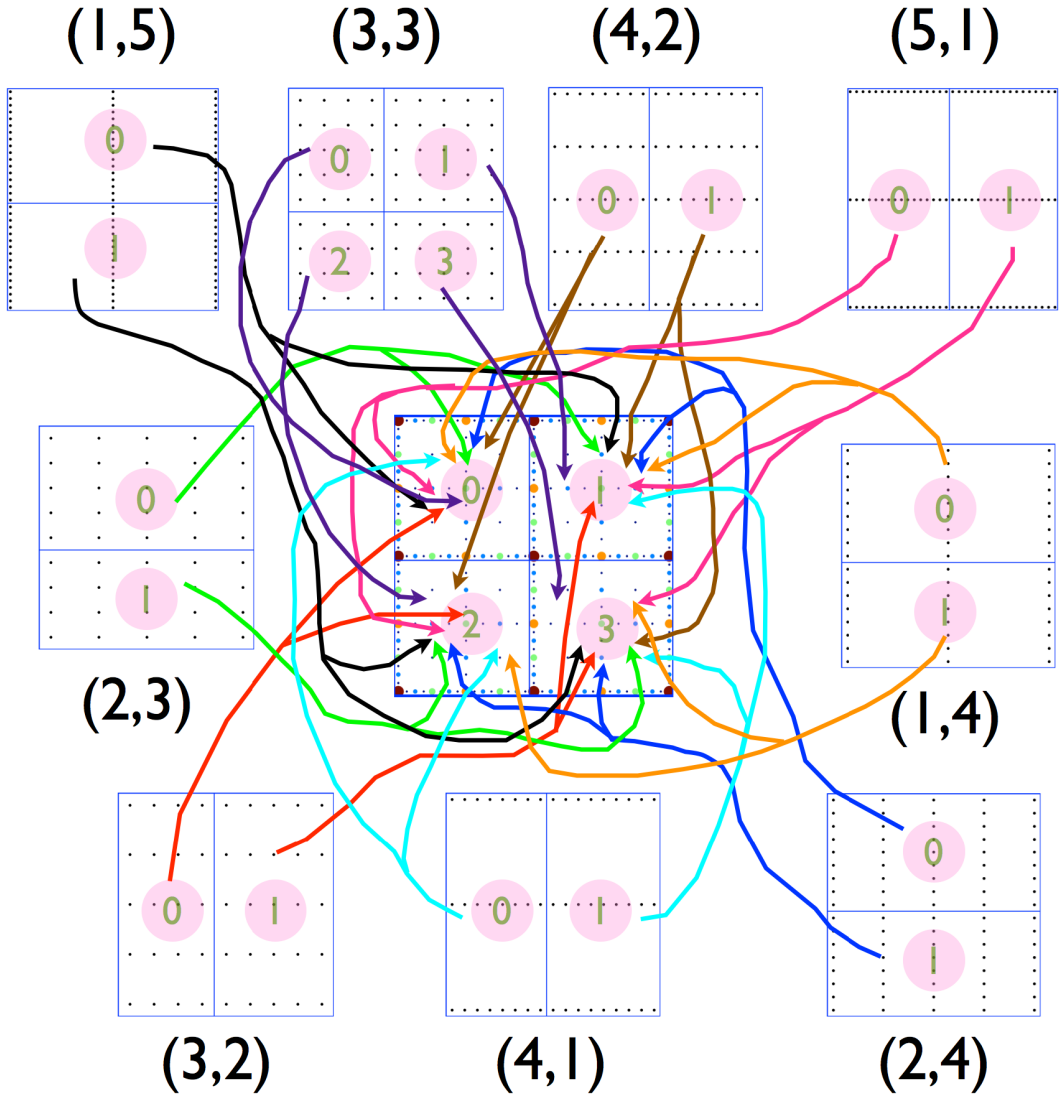


6 Robust Combination Techniques

- 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



7 Parallel SGCT Algorithm: the Gather-Scatter Idea



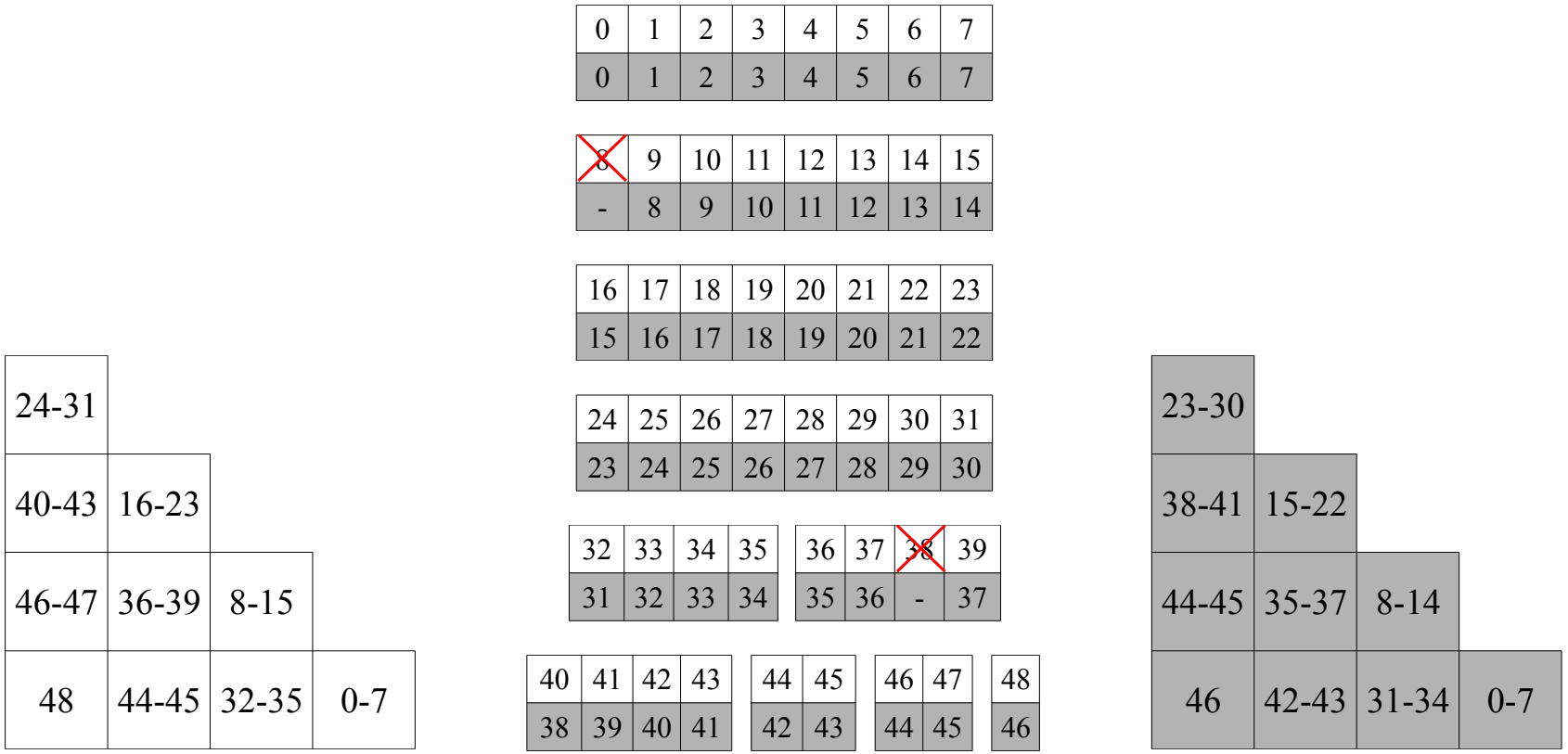
- evolve independent simulations over set of component grids, solution is a d-dimensional field (here d=2)
- each grid is distributed over a process grid (here these are 2×2, 2×1 or 1×2)
- gather: after a simulated time T is reached, combine fields on a sparse grid (here level 5, or index $(5, 5)$)
-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

8 Shrinkage-based Recovery of FT SGCT Applications

- each sub-grid is solved over a set of processes (with contiguous MPI ranks within the global communicator)
- we check for 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

9 Shrinkage-based Recovery of a $l = 4$ FT SGCT Application



(a) process sets before shrinking communicator

(b) details before & after

(c) process sets after shrinking communicator

10 Communicator Recovery via ULFM MPI

- recovery via process shrinkage similar to process replacement (see PDSEC'14 paper)
- create an ULFM MPI error handler, passing address of global communicator `ftComm` to it
- e.g. processes 3 and 5 of ranks 0–6 will 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 holding the MPI rank or comm. size

11 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, MPI rank of the first process
  - $P = (P_x, P_y)$, the process grid shape. Initially $n = P_x P_y$

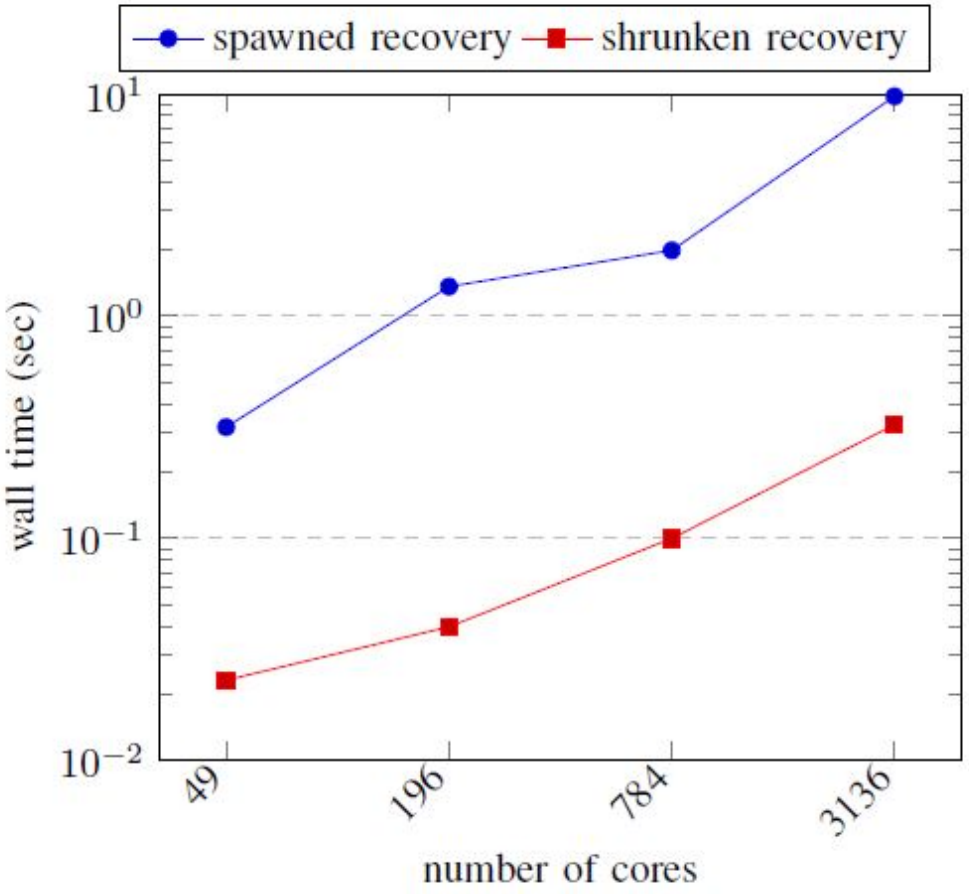
  A logical process id $p = (p_x, p_y)$, $(0,0) \leq p < P$, has rank $r_0 + p_y P_x + p_x$
- if this grid is numbered $i \geq 0$, $r_0 = \sum_{j=0}^{i-1} n_j$, where n_j is number of processes in grid j
- if we detect f failures in this grid, we resize to $P \leftarrow (P_x - \lceil f/P_y \rceil, P_y)$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

12 Modifications to the PDE Solver

- the initialization of all process grid dependent variables and arrays are put into a single function
  - note that a 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

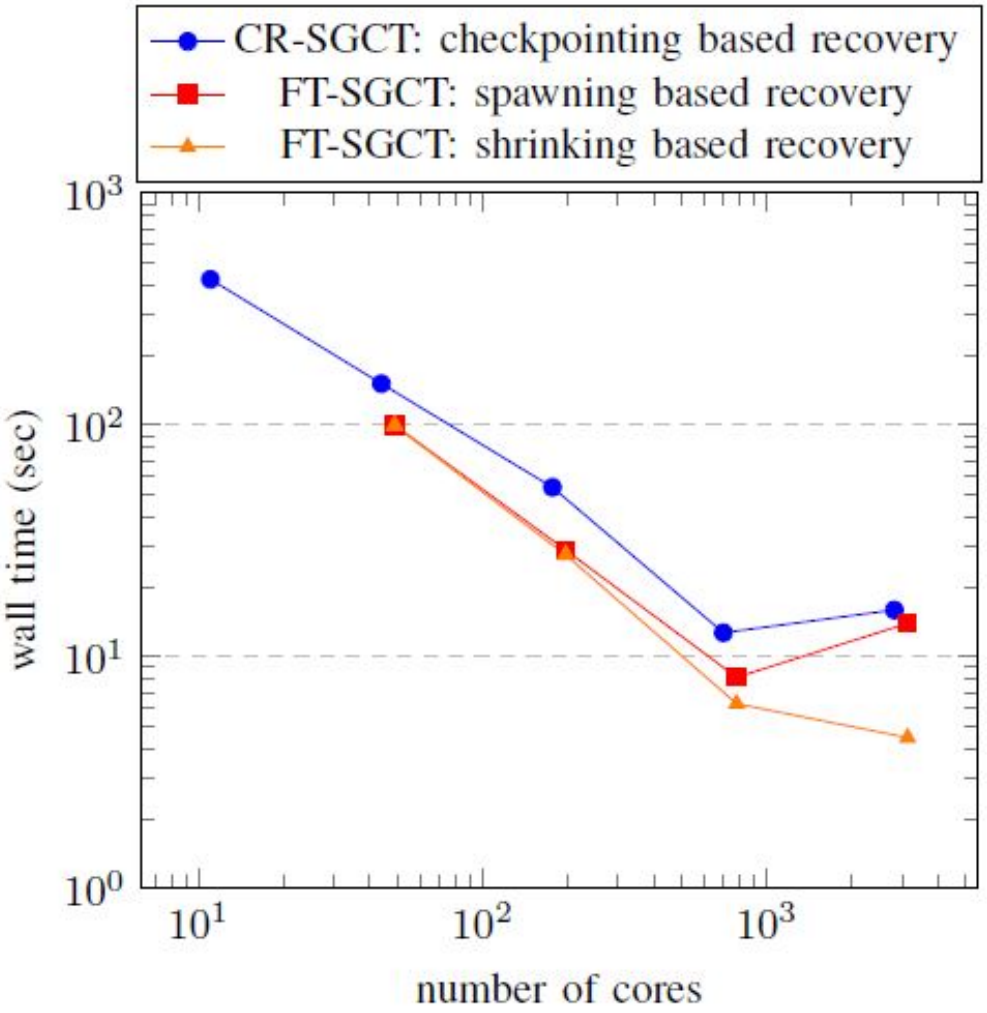
13 Results: Replace vs Shrink Recovery Overheads

- compare overheads of process replacement ('spawn') vs process shrinkage
- experiments on 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



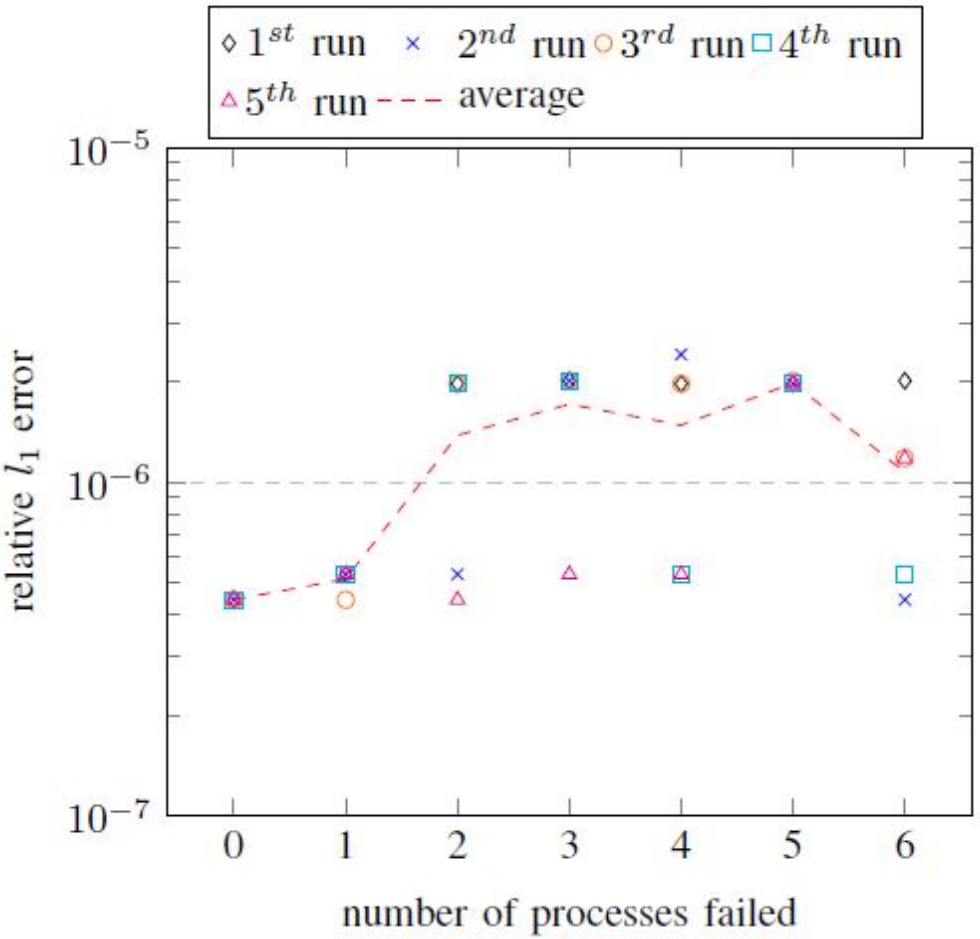
14 Results: Advection Application Performance

- 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 $\approx$ 3000 cores
- shrinkage fastest despite loss of compute resources



15 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 depended on which sub-grids had the failed processes
- note: results identical for replacement or shrinkage recovery



16 Conclusions: Part 1

- 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 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 agreement algorithm available
- advection solver accuracy still high even with $\approx 10\%$ process failures

17 Part 4: Robust Stencils – Motivations for Soft Faults

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

18 Finite Difference Computations

- finite differences methods and method of lines are common for solving partial differential equations
- explicit methods cannot leverage fault tolerant linear algebra
- Triple Modular Redundancy (TMR) could easily be used
  - do everything 3 times (with separate memory)
  - choose result which is equal for any two
  - 1/3 efficiency (3 times the resources/time)
- how else could we detect/correct or even avoid errors?

19 Robust Stencils in 1D

- $\delta_t u + a\delta_x u = 0$
- the standard LaxWendroff method $u_t^{n+1} = \frac{c(1+c)}{2}u_{i-1}^n + (1-c^2)u^{n-1} + \frac{c(-1+c)}{2}u_{i+1}^n$
  with $c = a\Delta t/\Delta x$ is stable and second order

  q
- Mayo et al. considered using several finite difference discretisations having distinct stencils to make computations fault tolerant

20 1D Robust Stenciks

- we can use a widened discretisation
  $u_i^{n+1} = \frac{c(2+c)}{8} u_{i-2}^n + \frac{4-c^2}{4} u_i^n + \frac{c(-2+c)}{8} u_{i-2}^n$
  and a third (far) stencil avoidng the centralo point:
  $un_{1i} = \frac{-3+8c+3c^2}{48} u_{i-3}^n + \frac{9-c^2}{25} u_{i-1}^n + \frac{9-c^2}{25} u_{i+1}^n + \frac{--8c+3c^2}{48} u_{i+3}^n$
  both of which are also stable and second order
- the corresponding stencils are:

21 Error avoidance and results in 1D

- to avoid (significant) errors:
  - take the median of $u_{i-1}^n, u_i^n, u_{i+1}^n$
  - discard the $u_{i-3}^n, u_{i-2}^n, \ldots u_{i+3}^n$ furtherest from the mediab
  - use the most cmpact stencil which avoids the discarded value

Results:

- similar robustness to TMR
- ISimilar efficiency to TMR (note: not yet optimised)
- similar ideas also applied to (inviscid) Burgers equation with similar success for robustness (note: shocks still captured

22 Advection in 2 or more dimensions

- we extend this work to 2D

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

  (NB: also applies to $d > 2$)

- consider a square domain discretised as a uniform grid
- the 1D discretisations can be applied in 2D by applying along one direction at a time (effectively an operator splitting approach)
- the resulting stencils are on the right

23 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 checkpoint)

Acknowledgments to:

- team members Markus Hegland and Brendan Harding
- ARC Linkage Grant LP110200410
- 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